UNIVERSITY OF GRANADA

DEPARTMENT OF COMPUTER SCIENCE
AND ARTIFICIAL INTELLIGENCE

PHD PROGRAM IN INFORMATION AND
COMMUNICATION TECHNOLOGIES

PhD THESIS DISSERTATION

Distance Metric Learning

for Explainability in Complex Problems

PhD CANDIDATE
Juan Luis Suárez Díaz

PhD ADVISORS
Salvador García López & Francisco Herrera Triguero

Granada, November 2023

Editor: Universidad de Granada. Tesis Doctorales
Autor: Juan Luis Suárez Díaz
ISBN: 978-84-1195-178-4
URI: https://hdl.handle.net/10481/89448

El doctorando / *The PhD candidate* **Juan Luis Suárez Díaz** y los directores / *and the advisors* **Salvador García López & Francisco Herrera Triguero**

Garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y, hasta donde nuestro conocimiento alcanza, en la realización del trabajo se han respetado los derechos de otros autores a ser citados cuando se han utilizado sus resultados o publicaciones.

Guarantee, by signing this doctoral thesis, that the work has been done by the doctoral candidate under the direction of the thesis supervisors and, as far as our knowledge reaches, the rights of the authors to be cited (when their results or publications have been used) have been respected in the performance of this work.

Granada, November 2023.

The PhD candidate:

Sgd.: Juan Luis Suárez Díaz

The PhD advisor:

Sgd.: Salvador García López

The PhD advisor:

Sgd.: Francisco Herrera Triguero

Funding

This doctoral thesis has been funded by the following research projects: the predoctoral scholarship FPU18/05989 given to the Ph. D. candidate by the Spanish Ministry of Science, Innovation and Universities, the national research projects TIN2017-89517-P and PID2020-119478GB-I00, and the regional research project A-TIC-434-UGR20.

«*Most people are not just comfortable in their ignorance, but hostile to anyone who points it out.*».

– Alex Gendler, interpreting Plato's Allegory of the Cave.

«*The question in each and every thing, 'Do you want this again and innumerable times again?' would lie on your actions as the heaviest weight!*».

– Friedrich Nietzsche, The Gay Science.

Agradecimientos

Hacer una tesis doctoral es un proceso largo y complicado en ciertos momentos, en el que se pasa por diferentes etapas: desde los momentos de mayor ilusión y motivación, como ese primer paper aceptado, viajes que jamás habría imaginado, o el día en el que tus esfuerzos como profesor son agradecidos; hasta los momentos más duros, cuando las cosas no salen, la ansiedad se apodera de ti o la soledad te inunda, todo ello acrecentado por una pandemia que estuvo presente durante los primeros años de esta tesis y que no facilitó las cosas. Es mucha la gente que ha pasado por mi vida durante estos años. Gente con la que he tenido la suerte de poder compartir los buenos momentos, y que me ha ayudado a superar los malos. Como persona de pocas palabras que soy, pocas veces he encontrado el momento de agradeceros a todos vuestro apoyo. Por ello, quiero aprovechar este momento para reflejar por escrito todo lo que siento.

En primer lugar, quería dar las gracias a mis directores, Salva y Paco, por quienes hoy puedo estar escribiendo este documento. A Salva le debo no solo la dirección de tesis, sino los buenos momentos que hemos vivido con conversaciones sobre la vida en general, los desayunos cuando empezábamos a retomar la normalidad tras el confinamiento, las tardes enteras jugando a juegos de mesa y su sinceridad y cercanía en general. A Paco le debo el haber depositado su confianza en mí desde mucho antes de comenzar la tesis, en los últimos años de carrera, y haberla mantenido hasta la actualidad. Siguiendo con la supervisión, también quería dar las gracias a Steven y Frank, por su acogida durante mi breve estancia en Cardiff, su apoyo y su paciencia, no solo durante la estancia, sino también en los meses posteriores.

Fuera del doctorado, no quería olvidarme de mis amigos de toda la vida. Mis compañeros del instituto son, posiblemente, los primeros a los que pude llamar amigos de verdad. Aunque ya no nos vemos mucho, seguimos manteniendo el contacto y casi todos los veranos o navidades conseguimos sacar un hueco para reunirnos todos y ponernos al día. Quiero acordarme aquí de Pedro, Natalia, Gema, Iván, Jorge, Félix y Agustín. En particular, Iván y Jorge, que fueron de mis mejores amigos en el instituto y que, por circunstancias de la vida, nos seguimos viendo con bastante frecuencia, y Félix y Agustín, también de mis mejores amigos, y con los que siento que tengo mucho en común, algo que no me suele ocurrir muy a menudo, y por los que estoy muy agradecido de que nuestros caminos se hayan cruzado y se mantengan cercanos pese a la distancia. Y no podía terminar este párrafo sin mencionar a dos de los mejores amigos que he hecho nunca, ya durante la carrera. En primer lugar, a

Andrés, a quien conozco desde las olimpiadas matemáticas del instituto, una de las personas de la que más he aprendido durante la carrera, y con quien he tenido la suerte de volver a coincidir en esta recta final del doctorado. Y en segundo lugar, a Moya, una persona genial, con la que he compartido muchos de los mejores momentos de la carrera y a la que echo mucho de menos desde que se fue a Córdoba.

Posiblemente a quienes más les tengo que agradecer estos años es a todos los amigos que me han acompañado, tanto en alegrías como en sufrimientos, haciendo el doctorado a la vez que yo. Tanto a los que ya dejaron la universidad, como José Alberto, Paco Luque, Laura, José Ángel, Sergio o Jesús, de los que guardo buenos recuerdos, como a los que vinieron en la recta final: los peques Víctor, Prá, Iván GPUs, Jota, Marina y Carlos, una generación llena de energía positiva y con la que me lo he pasado muy bien en esos ratos de comedor, cafetería, descansos, pádel y celebraciones. Y, por supuesto, a los que han estado ahí desde el principio o casi y con quienes he compartido la mayor parte de estos cuatro años: Javi, a quien conozco desde primaria y que ha sido compañero de sufrimiento en los proyectos con empresas; Elena, compañera de carrera y de las personas más alegres que he visto en el ámbito del doctorado; Guille, la voz de la experiencia, compañero de docencia y de quien he aprendido mucho, especialmente en esta etapa final; José Daniel, el amigo de confianza con quien comparto mi pasión por los videojuegos; Iván, un gran amigo desde el instituto, durante la carrera y hasta hoy, que siempre trae diversión y buen humor; Nacho, esa persona que siempre está ahí cuando tienes algún problema, un gran descubrimiento y una fantástica persona; Germán, compañero administrador de Hércules, con quien me pasé días en la sala de servidores tras el confinamiento para hacer funcionar el nuevo cluster, la persona que simbolizó para mí el fin del confinamiento y con la que he compartido uno de los viajes más espectaculares que he hecho nunca; y Nuria, compañera de carrera y de docencia, a quien le quiero dedicar más adelante varias líneas.

También en esta etapa de doctorado quería acordarme de dos personas que, sin pertenecer a mi entorno, me han aportado mucho, cada una a su manera. Por un lado, mi psicóloga Ana, que me ha ayudado a superar los momentos difíciles que han ido surgiendo en esta etapa. Y por otro lado, Javier León, cuyas revisiones de inglés me han ayudado mucho en los artículos y trabajos que he escrito para esta tesis.

Una de las cosas que más he disfrutado durante estos cuatro años es el haber tenido la oportunidad de dar docencia en la universidad. Por este camino también he conocido a gente maravillosa a la que le debo mucho. En primer lugar, a mis compañeros de la asignatura de Inteligencia Artificial, Juan, Raúl y Antonio por su apoyo constante y disposición a intentar mejorar la asignatura siempre que les hemos propuesto algo. He aprendido mucho gracias a ellos. También de Cristina, con quien he compartido dos años de prácticas y que ha hecho que mi etapa como profesor sea muy amena. Y, por supuesto, no me podía olvidar de Nuria. Compañera de prácticas durante los tres años que he sido docente, la mejor compañera que he podido tener. Y no solo eso, también ha sido mi compañera durante toda la carrera y de las mejores amigas que he tenido nunca. Con quien se puede contar para todo, y quien con su energía siempre nos hace más felices a todos los que estamos a su alrededor. Gracias por todo.

También gracias a la docencia, he tenido la oportunidad de dar clase a gente fantástica. Quería acordarme aquí de todos los alumnos y alumnas que he tenido durante estos años, que han hecho que la docencia sea una de las experiencias más gratificantes que he vivido nunca. A los alumnos de mi primer año quiero agradecerles su comprensión y paciencia, más aún con las complicaciones de la vuelta a las clases tras la pandemia. Con los de mi segundo año viví mi primera experiencia presencial dando clase, y gracias a ellos pude sentirme plenamente profesor por primera vez. Y a los de mi tercer año quiero agradecerles su alegría, ilusión y cariño, ya que con ellos me he sentido realmente feliz como profesor, y me han hecho plantearme de verdad seguir como docente en el futuro. Muchos nombres han pasado estos años por mis clases, y quería destacar a varios en especial: Gabriel, Paco, Álvaro, David, Laura, Pablo, Carmen, ... Me habéis hecho muy feliz. Y por último, pero no por ello menos importante, María. No podría haberme alegrado más de haberte conocido, y de lo feliz que me has hecho durante los últimos meses de mi doctorado. Gracias por haber aparecido en mi vida.

Y, para terminar, no podían faltar en esta sección quienes me han acompañado durante toda mi vida: mi familia. Quería dar las gracias a mi madre y mi padre, Esther y Juan Luis, por estar siempre ahí. Gracias a vosotros he podido llegar hasta aquí. A mi hermano Jorge, con quien he compartido la mayor parte de mi vida. A mis abuelos, Gilberto y Manuel, que hace ya tiempo que no están con nosotros, pero que siempre estarán en mi memoria. A mis abuelas, Inés y Marina, y mi tía Amalia, por su cariño infinito. Y al resto de mis tíos, tías, primos y primas. Gracias a todos.

No habría llegado hasta aquí sin todo el apoyo que he recibido. Tanto a los que he mencionado en este texto como a los que me he tenido que dejar: a todos los que me habéis acompañado en esta etapa,

Gracias.

Abstract

The major technological advances of recent years have led to the generation of large amounts of data from which, if properly processed and analyzed, relevant information can be extracted for many fields such as science, business or communication. Machine learning, which is part of the process known as *knowledge discovery in databases*, is emerging as a discipline focused on the study of techniques that allow machines to learn from data, in the sense of recognizing patterns or drawing inferences from previously unseen data.

Within machine learning, there is a set of algorithms called similarity-based learning algorithms, which are inspired by one of the most powerful mechanisms of human learning: recognizing objects based on their similarity to other previously seen objects. These algorithms require a distance or similarity measure between the data, which allows us to assess the similarity of any two samples. This distance or similarity measure is crucial for the correct functioning of these algorithms, since the quality of the results obtained depends on it.

In this thesis, the *distance metric learning* problem is addressed. This problem consists in learning the distance or similarity measures from the data itself, so that they can be successfully used later in similarity-based learning algorithms. Specifically, this project tackles the study and development of distance metric learning algorithms and their application in novel or uncommon problems of machine learning, beyond the classic classification or regression problems.

This thesis addresses the following objectives:

1. The study of distance metric learning and its algorithms from both a theoretical and an experimental point of view. To this end, the development of a software library with the highest-performing algorithms in the field is proposed, as well as a tutorial that includes a theoretical review, an experimental study and an analysis of the results of the algorithms.

2. The development of new distance metric learning algorithms for unconventional or singular problems, i.e., machine learning problems beyond the classic standards of classification or regression. To achieve this goal, three distance metric learning algorithms have been developed and proposed for the first time in this thesis, which

address three different singular problems: imbalanced, ordinal and monotonic classification.

3. The development of deep metric learning models to tackle complex problems. In recent years, deep learning has revolutionized the field of machine learning, and this revolution has also reached distance metric learning. Deep metric learning proposes new models for learning distances that open up a new range of possibilities in this field. In addition, these methods have proven to be effective in some of the most challenging problems in deep learning, such as those where little data is available. In regard, this thesis includes a proposal for a deep metric learning model applied to a natural language processing problem with data scarcity.

4. The analysis of the explainability of the developed models, based on the explainable characteristics of the similarity-based learning algorithms, and on how learning a distance influences these characteristics.

5. The specialization of the developed proposals for their application in real problems. This is addressed together with the third objective in the natural language processing problem treated.

This thesis project successfully addresses the above objectives and thus leaves notable contributions in the field of distance metric learning. The software library and tutorial developed here provide a solid basis for understanding the state-of-the-art in traditional distance metric learning, and a practical starting point for those who want to enter the discipline. The algorithms proposed in the second objective provide new perspectives to address less common problems in machine learning; the deep metric learning models developed in the third objective show the potential of combining deep learning and distance metric learning, and are also applied in a real case study, thus addressing the fifth objective. Finally, the explainability study proposed in the fourth objective analyzes, for one of the developed algorithms, how distance metric learning can influence the explainability of the resulting model.

Resumen

Los grandes avances tecnológicos de los últimos años han traído consigo la generación de grandes cantidades de datos, de los cuales se puede extraer información relevante para numerosos campos, como la ciencia, los negocios o la comunicación, si se procesan y analizan adecuadamente. El aprendizaje automático, englobado dentro del proceso conocido como *knowledge discovery in databases*, surge como disciplina centrada en el estudio de técnicas que permitan que las máquinas puedan aprender a partir de los datos, en el sentido de reconocer patrones o hacer inferencia sobre datos no vistos previamente.

Dentro del aprendizaje automático, hay una rama de algoritmos denominados de aprendizaje basados en semejanza, que se inspiran en uno de los mecanismos más potentes del aprendizaje humano: el reconocimiento de objetos basado en la similitud con otros objetos previamente vistos. Estos algoritmos requieren de una medida de distancia o de similitud entre los datos, que permita determinar cómo de parecidos son cualesquiera dos ejemplos de los que se disponga. Esta medida de distancia o similitud es crucial para el correcto funcionamiento de estos algoritmos, ya que de ella depende la calidad de los resultados que se obtengan.

En esta tesis se aborda el problema del *aprendizaje de métricas de distancia*, que consiste en aprender las medidas de distancia o similitud a partir de los propios datos, de forma que puedan ser empleadas posteriormente en algoritmos de aprendizaje por semejanza de forma exitosa. En concreto, se afronta el estudio y desarrollo de algoritmos de aprendizaje de distancias y su aplicación en problemas novedosos o poco comunes del aprendizaje automático, más allá de los problemas clásicos de clasificación o regresión.

Esta tesis aborda los siguientes objetivos:

1. En primer lugar, se propone abordar el estudio del aprendizaje de distancias y sus algoritmos tanto de un punto de vista teórico como experimental. Para ello, se plantea el desarrollo de una librería software con los algoritmos más destacados de la disciplina, y un tutorial que incluya una revisión teórica, un estudio experimental y un análisis de resultados de los mismos.

2. El segundo objetivo plantea el desarrollo de nuevos algoritmos de aprendizaje de distancias para problemas no convencionales o singulares, es decir, problemas del apren-

dizaje automático que se salen de los estándares clásicos de clasificación o regresión. Para cumplir con este objetivo, se han desarrollado tres algoritmos de aprendizaje de distancias, propuestos por primera vez en esta tesis, que abordan tres problemas singulares diferentes: clasificación desbalanceada, ordinal y monotónica.

3. El tercer objetivo se centra en el aprendizaje de distancias profundo. En los últimos años, el aprendizaje profundo ha revolucionado el campo del aprendizaje automático, y esa revolución también ha llegado al aprendizaje de distancias. El aprendizaje de distancias profundo propone nuevos modelos para aprender distancias que abren un nuevo abanico de posibilidades en este área. Además, estos métodos han demostrado ser efectivos en algunos de los problemas más desafiantes del aprendizaje profundo, como aquellos en los que se dispone de pocos datos. Esta tesis incorpora una propuesta de modelo de aprendizaje de distancias profundo aplicada a un problema de procesamiento del lenguaje natural con escasez de datos.

4. El cuarto objetivo es transversal y plantea el análisis de la explicabilidad de los modelos desarrollados, apoyándose en las características explicables de los algoritmos de aprendizaje por semejanza, y en cómo aprender una distancia influye a dichas características.

5. Por último, se plantea la especialización de las propuestas desarrolladas para su aplicación en problemas reales. Esto se aborda conjuntamente con el tercer objetivo en el problema de procesamiento del lenguaje natural tratado.

La tesis aborda con éxito los objetivos enumerados, dejando así aportaciones destacables en el campo del aprendizaje de métricas de distancia. Con la librería software desarrollada y el tutorial se proporciona una base sólida para comprender el estado del arte del aprendizaje de métricas de distancia tradicional, y un punto de partida para iniciarse en la disciplina de forma práctica. Los algoritmos propuestos en el segundo objetivo proporcionan nuevas perspectivas para abordar problemas menos habituales del aprendizaje automático, y los modelos de aprendizaje de distancias profundo desarrollados en el tercer objetivo muestran el potencial de combinar aprendizaje profundo y aprendizaje de distancias, siendo aplicados además en un caso de estudio real, abordando así el quinto objetivo. Por último, el estudio de la explicabilidad propuesto en el cuarto objetivo analiza, para uno de los algoritmos desarrollados, cómo el aprendizaje de distancias puede influir en la explicabilidad del modelo resultante.

Table of Contents

List of Abbreviations

AI	Artificial Intelligence
CBR	Case-Based Reasoning
C-Index	Concordance Index
CMOML	Chain Maximizing Ordinal Metric Learning
CNCA	Condensed Neighborhood Component Analysis
DA	Domain Adaptation
DML	Distance Metric Learning
GNN	Graph Neural Network
FSL	Few Shot Learning
KCMOML	Kernel Chain Maximizing Ordinal Metric Learning
KDD	Knowledge Discovery in Databases
***k*-NN**	*k*-Nearest Neighbors
LM^{3}L	Large Margin Monotonic Metric Learning
MAE	Mean Absolute Error
ML	Machine Learning
NCA	Neighborhood Component Analysis
NCM	Nearest Class Mean
NLP	Natural Language Processing
NMI	Non-Monotonic Index
NN	Nearest Neighbors

NOTA	none-of-the-above
RE	Relation Extraction
SVM	Support Vector Machine
XAI	Explainable Artificial Intelligence

Chapter I

PhD Dissertation

«*Quiet people have the loudest minds.*».
– Stephen Hawking.

1 Introduction

The content of this Ph.D. dissertation is inspired by one of the most important cognitive mechanisms of human learning. Let us imagine that we are visiting a friend's house for the first time, and he shows us his living room. Immediately, we begin to recognize all the objects we see: a table, a chair, a sofa, a television, a telephone, ... All this even though we have never been there before. How are we able to do this? How do we know that the table is indeed a table and not some other object? The answer to these questions is that we are able to recognize the objects by their similarity to other objects that we have seen before. That is, because we have seen tables, chairs, televisions, telephones, etc. in other places before. And we may have never seen exactly the same table, or the same telephone, but we have seen objects similar enough to be able to determine that they are of the same type. Although we take this process for granted and do not usually think about it consciously, it is essential to our learning from birth.

We can take this a step further. Let us imagine that we are shown several pictures of animal species that we have never seen before. We immediately realize that we do not know what they are. We may think they are similar to some species we have seen before, but we recognize that they are not similar enough to be considered the same species. That is, we are able to determine that it is an animal unlike any other we have seen, even though we cannot say exactly what it is. Not only that, but once we are told what the animal is, we may be able to recognize it as soon as we see a picture of it again. That is to say, once we have learned this concept, we are able to identify it every time it is presented to us. And the most natural fact is that, in order to learn it, we only need to be told about that concept once, or a few times at most.

If we transfer this idea to the field of technology, one of the first questions that may come to mind is: can we make a computer learn by similarity, as we humans do? More specifically, when a device receives input data of any kind, is it possible to make it capable of recognizing what it is based on similarity to other data it has previously received?

The first thing we need to consider is how to handle the input data. These data can be practically anything we can think of, depending on the problem we are trying to solve: images, text, metrics, colors, surveys, etc. It is necessary to extract the relevant information from these data in a way that can be processed by a computer. In general, from any type of data we will be able to extract a set of features that can be encoded as a numerical vector. So, once we have these numerical data, can we establish a similarity measure among them?

The answer is yes. Once we have our vectors or points in the plane, in space, or in any higher dimension, we can measure distances among them, and consequently say that two instances are similar if they are close, and different if they are far. Then, the data that make up the "memory" of our device will contain all the previous experiences learned and, when new data are received, we can compute its similarity to the stored data and, based on that, determine what kind of data it is.

Figure 1 shows an example of this learning mechanism. Let us imagine that our data are

images of animals, which we have managed to transform into two-dimensional numerical data—as shown in the figure—and we want our machine to learn to recognize which animal it is each time we receive a new image of an animal. Already in the memory of our device we have the colored dots representing different types of animals: dog (blue), cat (red) and horse (green). And now we have three new images, (1), (2) and (3), and we want to find out what type of animal they represent. The vector associated with image (1) is surrounded by blue points, which means that it is similar to the data of type "dog" we had stored previously, and quite different from the other two types, since it is far from them. Therefore, our machine can conclude that example (1) is a dog. Example (2) is close to the red points, but also close to one of the blue dots. Possibly because of the shape or colors of the image, our engine finds similarities with both the cat images and the dog images we had stored. In any case, the vector associated with example (2) is somewhat closer to the red points, so example (2) would ultimately be classified as a cat. Finally, example (3) is isolated from all the others. This situation would be analogous to being shown a picture of an animal we have never seen before: it does not resemble anything we have previously encountered, so we cannot tell what animal it is.

Although the above example seems to mimic the human similarity learning mechanism well, it has two drawbacks. First, we do not know how the data represented in the space of Figure 1 were obtained from the images. Even if we assume that the data represent the most relevant information from the images, we have no guarantee that, just by being represented in the plane, the most similar images must be closer together. Or at least close in the sense we assume, which is related to the second drawback: in Figure 1 we have assumed that the concept of "close" is given by the usual distance in the plane, the *Euclidean distance*. But what if this distance is not always the right one? Let us have a look at Figure 2. In this case, we see that the data are arranged—strangely, but there is a clear pattern. The relevant information seems to be represented in the vertical coordinate, which determines what type the given examples will be (A, B or C). But, in this case, the new samples (marked with '?') are misclassified because they are closer to examples with a different vertical coordinate, and the Euclidean distance cannot extract such information. In general, we will have a myriad of ways to measure distance, and it is very likely that Euclidean distance is not the most appropriate in many cases. This is the point of divergence with respect to human learning. While we have internalized a virtually infallible similarity measure for any pair of objects presented to us, finding an appropriate similarity measure for a given problem is one of the major challenges of machine-level similarity-based learning.

Just as in Figure 1 we used the stored data to try to identify new data, it may be natural to ask the following question: what if an appropriate distance can also be found by learning it from the data themselves? Since ideally data of the same type should be close and data of different types should be far apart (which is referred to as the *smoothness assumption*), why not extract from the data themselves what must be "close" and what must be "far apart"? This is the basis of *distance metric learning*, which we will formalize later. The purpose of this Ph.D. thesis in the research area of Data Science is to deepen this discipline with the development of models to solve different problems in the field of Machine Learning and

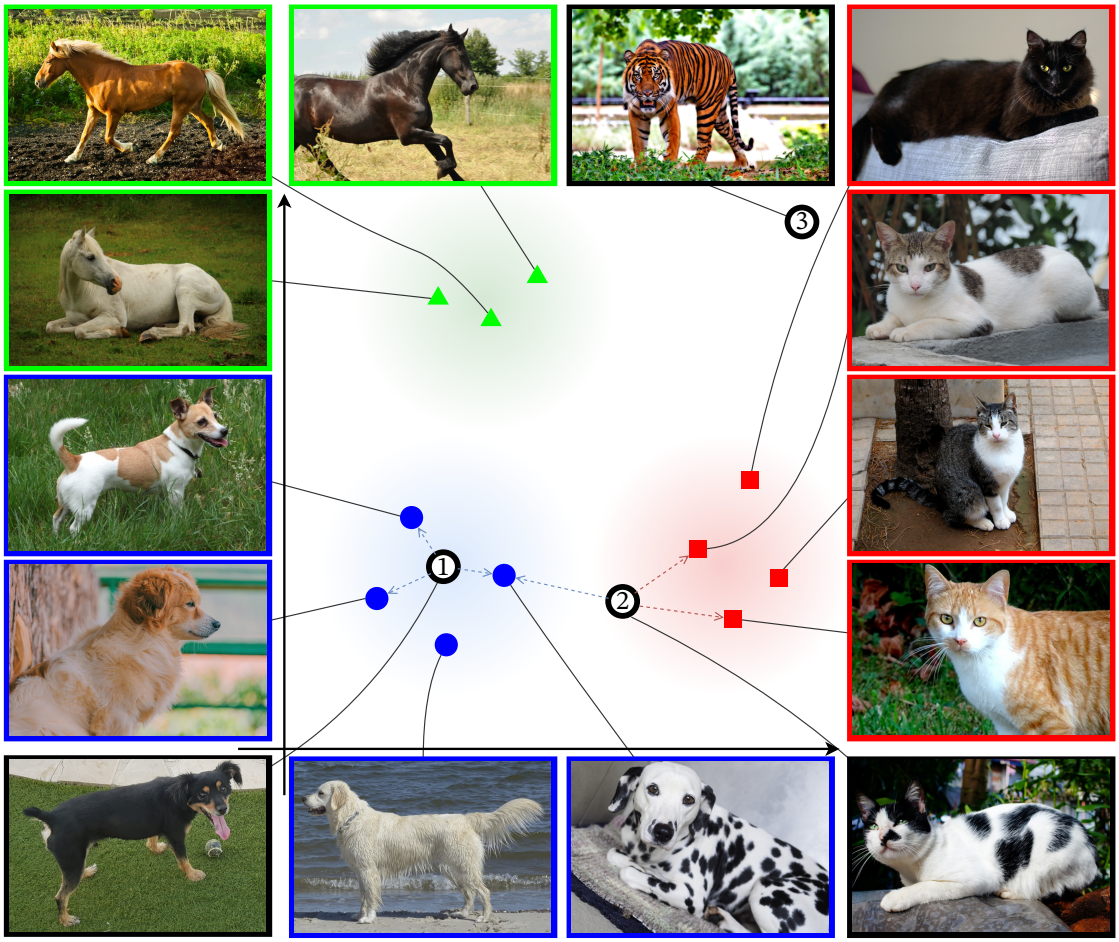


Figure 1: Similarity-based learning of animal species.

Data Mining.

In today's digital landscape, the amount of data generated, collected and stored around the world has grown exponentially. This vast ocean of information holds valuable insights and opportunities that can transform industries, improve decision-making, and drive significant advances in a wide range of fields. However, managing, analyzing and extracting useful information from this overwhelming amount of data poses significant challenges. This is where the process of Knowledge Discovery in Databases (KDD) [PF91] comes in to transform data into valuable information and, ultimately, useful knowledge.

KDD is an interdisciplinary field that combines data mining, machine learning, statistics and database techniques to discover hidden patterns, trends and relationships in data. The process of data mining can be divided into several stages [MR05]: problem specification, data extraction and selection, data preprocessing and transformation, data mining, interpretation and evaluation. Machine Learning (ML) [SSBD14] is the branch of Artificial In-

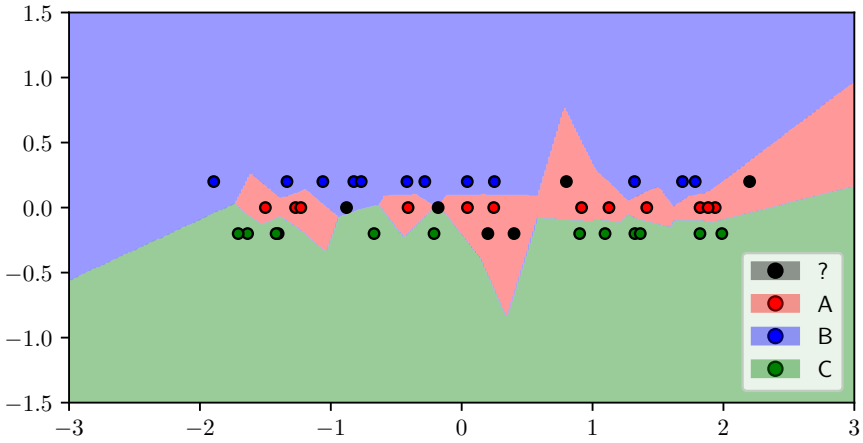


Figure 2: When the Euclidean distance is not the right one.

telligence (AI) concerned with the design and development of algorithms and models that enable computers to detect relevant patterns in data, and it is of great importance in many stages of the KDD process. Especially in data mining [HK06, A+15], responsible for extracting knowledge from data, with the help of ML techniques, but also in the preprocessing and transformation stage [GLH15], where such techniques can be used to detect noise, fill incomplete data, select relevant features and, in general, transform the data to make it of higher quality [LGGRG+20] for the data mining stage.

Within ML techniques, *similarity-based learning* is based on the idea that data of the same type must be similar—in the sense that their representation in the feature space must be close—according to some measure of similarity or distance that is defined. This type of algorithm can be applied to a variety of learning problems of general interest. For example, given a set of labeled data, we can try to predict the labels of incoming data according to their similarity to the data we already have. Or, given an unlabeled dataset, we can try to find appropriate groupings of data of the same type based on the similarity among them. Similarity-based learning techniques have been around since the early days of ML; in fact, one of the best known algorithms is the k-Nearest Neighbors (k-NN) classifier [CH67], which, given a set of labeled data and a sample to label, assigns to that sample the majority label among its k nearest neighbors within the labeled data, following the idea shown in Figures 1 and 2.

In order for similarity-based learning techniques such as k-NN to work, it is necessary to establish a measure of similarity or, equivalently, a measure of distance, among the data. Typically, standard distances, such as the Euclidean distance, are used when working with numerical data. However, these standard distances may not be the most appropriate for the data we are working with because they do not take into account the underlying characteristics of the dataset. Distance Metric Learning (DML) [XJRN03] was created with the purpose of learning distances from the data itself, so that they can later be used by similarity-based learning algorithms to achieve better performance. Thus, it can be considered as a set

of data preprocessing and transformation techniques to improve the quality of the data for subsequent similarity-based learning.

There are three main paradigms in machine learning: supervised, unsupervised and reinforcement learning. DML, together with similarity learning, encompasses a wide variety of techniques of different nature, having scope in all three paradigms:

- **Supervised learning:** the goal of this paradigm is to learn a function to predict the labels of incoming data, given a training set of labeled data. Depending on the nature of the labels, there are two main types of problems: *classification* [DH+06], where the labels take a finite set of values, and *regression* [DS98], in which the labels take continuous values. DML can be used in supervised learning to bring data of the same class or with close labels closer together and push those of different classes farther apart. This can improve the subsequent performance of similarity-based classifiers or regressors, such as k-NN.
- **Unsupervised learning:** in this paradigm there are no labeled data, and the goal is to find patterns or structures. This paradigm consists of problems such as *clustering* [Mir12], which consists in finding clusters of data that are similar. Again, the similarity among the data is critical, so similarity-based learning algorithms like k-means [M+67] are very useful in this problem. DML can be very effective in this problem if additional information about the data is available, as we will see below. On the other hand, DML is a powerful tool in another unsupervised problem: *dimensionality reduction*. This problem consists in reducing the number of variables in the data while retaining as much relevant information as possible. Its goals include cleaning the data and reducing the computational cost of learning algorithms. The arrangement of the data in space and the distances among them are fundamental to find a suitable projection of the data, so DML techniques can be used to find such a projection by relying on the distances.
- **Reinforcement learning:** in this paradigm, the goal is to learn a policy that maximizes the long-term reward obtained by an agent in an environment [KLM96]. The agent interacts with the environment by performing actions, and the environment responds with a reward and a new state. DML can be used in this paradigm to learn a distance metric that allows the agent to encode meaningful representations of the states by creating a feature space in which the states that are similar are close together [TKS11].

DML algorithms have been widely used to enhance similarity-based learning algorithms in traditional problems such as standard classification, regression or clustering [WS09, GHRS05, MVPC13, XJRN03, WT07]. The first objective of this thesis is to study these algorithms and their replication at software level, with the aim of deepening the knowledge of this discipline and providing a framework for their experimental application.

However, the technological development of recent years poses new challenges, either because of the emergence of new learning problems for which it is necessary to adapt tra-

ditional techniques, or because of the new tools available to solve them. As for the new problems that arise, in this thesis we highlight several variants of the supervised learning problem:

- **Semi-supervised learning:** it includes problems where partial supervision is available [VEH20], which can be expressed in several ways. On the one hand, we may have two subsets of data, one labeled and one unlabeled; the goal is to take advantage of the unlabeled data—which are usually much more numerous because they are less expensive to obtain—to improve classification or regression accuracy. The propagation of information from labeled to unlabeled data can be done by relying on the similarity of the data, so again DML comes into play in this problem [ZG02]. On the other hand, we could have *weak supervision*, which means that, even if we do not know the labels of the data, we know that there are samples that must be of the same type and samples that must be different. DML can be used to bring together samples that must be similar and separate those that must be different. This can be applied to problems such as *constrained clustering* [GAPDP+23], where the goal is to find appropriate clusters, knowing that there are samples that must be in the same cluster and others that must be in different clusters.
- **Imbalanced classification:** in this classification variant, the unevenness in the number of samples in each class results in one or more majority classes and one or more underrepresented minority classes [FGG+18]. This often results in learning algorithms being biased towards the majority classes that in turn causes poor performance in the minority classes. Just as certain similarity-based techniques have been used to address the problem [Har68], the simultaneous use of these techniques and DML can be of great help in achieving substantial improvements in the problem.
- **Ordinal regression:** in this variant, the data labels take values from an ordered set [GPOSM+16], such as the stars of product ratings or the severity levels of a disease. In this case, it is the similarity among the labels that is relevant, since the closer two values are in the labels, the more similar their inputs should be. Incorporating this behavior into a DML technique can significantly improve the performance of similarity-based ordinal classifiers.
- **Monotonic classification:** in this ordinal regression problem, the input data also have ordering constraints [CGK+19]. For example, in a house pricing prediction problem, if all the attributes of one house are superior to those of another house in the same neighborhood, it is logical that the price of the former will be higher than that of the latter. As in ordinal regression, DML can be useful, but in this case the distance metric must take into account the monotonicity constraints of the data.

These problems are of interest in many real-world applications such medicine [FDA23, HLW+16], industry [YWH+23], economics [LPC+23, CL14, ZG22] or anomaly detection [TLL22]. The second objective of this thesis focuses on proposing models for these types of problems and evaluating their performance.

Moreover, among the new tools that have emerged from the technological advances of the last few years, it is worth highlighting *deep learning* [GBC17], which has revolutionized the field of ML. It is based on the use of deep neural networks, which are able to work with a wide variety of data types and automatically learn representations of the data. These representations are typically of high quality and, as a result, deep neural networks have achieved outstanding results on a broad range of ML problems. However, deep learning also brings along new challenges; most notably:

- **The need for large amounts of data to learn:** deep neural networks have millions of parameters to learn and therefore need large amounts of data to tune them. *Few Shot Learning (FSL)* [WYKN20] is a discipline that attempts to address this problem: in FSL problems, a very small training dataset is available, and the goal is to learn to classify new data with those few samples—and possibly with a larger auxiliary dataset of a different type from that of the one being learned.
- **The lack of transparency in the models:** deep neural networks are very complex models, so it is nearly impossible to understand how they work and why the make the decisions they do. This is a problem in many applications, such as medicine, where trust in the model ultimately depends on the reasons behind its decisions. The development of an Explainable Artificial Intelligence (XAI) is an emerging research field that aims to address this problem [ADRDS+20], not only in deep neural networks but in any ML model in general. XAIs are part of a more general concept called trustowrhy AI. Trustworthy AI encompasses a broader set of principles and practices to ensure that AI systems are robust, legal and ethical [AAES+23]. One of the requirements to achieve this goal is model transparency, defined as the ability for a model to be understood by a human in a natural way. And a fundamental component of transparency in a model is that it is able to explain the decisions it makes [DRDSC+23].

In recent years, DML models using deep neural networks internally have been proposed, giving rise to a new learning paradigm: *deep distance metric learning* or *deep metric learning* [KB19]. Deep metric learning seeks to combine the learning potential of deep neural networks with the smoothness assumption DML is inspired by in order to learn models that are able to represent data optimally from a similarity point of view and with higher performance than classic DML models. Deep metric learning models differ from deep learning models in that the latter learn to recognize the different classes in the data, while the former learn to discriminate between samples of different types without the need to explicitly recognize each class present in the problem, which has proven to be very useful in problems such as FSL [LYMX23]. The development of deep metric learning models and their application to complex deep learning problems such as FSL is the third objective of this thesis.

Finally, two further objectives are addressed in this thesis. First, as already mentioned, the development of learning models that are explainable is of interest in numerous real-world applications. The fourth objective of this thesis deals with the study of explainability in DML models and posterior similarity-based learning, and the interaction between the

two. The last objective of the thesis explores the application of the developed models in real problems.

To conclude this introduction, we give a summary of the structure of this thesis. It is made up of three chapters: the Ph.D. dissertation (Chapter I), the publications that support the knowledge and conclusions presented in the dissertation (Chapter II) and the work in progress that completes the remaining objectives proposed for the thesis (Chapter III). The dissertation consists of this introduction and the following sections: Section 2 elaborates on the theoretical foundations and concepts used throughout the thesis; Sections 3, 4 and 5 detail the justification, objectives and methodology on which this thesis is built, respectively; a summary of the research performed is presented in Section 6, and the results obtained are discussed in Section 7; finally, Section 8 discusses the conclusions of this thesis and future lines of research.

Chapter II contains the publications that support the knowledge and conclusions presented in the dissertation. Four publications have been included: three have been published in international indexed journals, and one is currently undergoing the second round of peer review. The publications are listed below:

- `pyDML`: A Python Library for Distance Metric Learning.
- A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges.
- Ordinal regression with explainable distance metric learning based on ordered sequences.
- Metric learning for monotonic regression: turning the space up to the limits of monotonicity.

Finally, Chapter III presents the current state of the last work planned for this thesis, which is still in progress. This work is written as a scientific paper and its content and structure are further detailed in that chapter.

Introducción

El contenido de esta tesis nace inspirado por uno de los mecanismos cognitivos más importantes del aprendizaje humano. Imaginemos que vamos de visita por primera vez a casa de un conocido, y nos enseña su salón. Inmediatamente, empezamos a reconocer todos los objetos que vamos viendo: una mesa, una silla, un sofá, una televisión, un teléfono, ... Y todo esto a pesar de no haber estado nunca allí antes. ¿Cómo somos capaces de hacer esto? ¿Cómo sabemos que la mesa es efectivamente una mesa y no cualquier otro objeto? La respuesta a estas preguntas es que somos capaces de reconocer los objetos por su similitud con otros objetos que ya habíamos visto previamente. Es decir, porque ya habíamos visto antes, en otros lugares, tanto mesas, como sillas, como televisiones, como teléfonos. Y puede que nunca antes hayamos visto una mesa exactamente igual, o ese teléfono, etc., pero sí hemos visto objetos lo suficientemente parecidos como para poder determinar que son del mismo tipo. Aunque este proceso lo tenemos más que asumido y normalmente no nos paramos a pensar en él, es esencial en nuestro aprendizaje desde que nacemos.

Podemos ir un paso más allá. Imaginemos que nos enseñan varias fotos de especies animales que nunca hemos visto antes. Inmediatamente nos damos cuenta de que no sabemos lo que son. Podemos pensar que son similares a alguna especie que sí hayamos visto antes, pero reconocemos que no lo suficiente como para que se puedan considerar del mismo tipo. Es decir, somos capaces de determinar que es un animal diferente a cualquier otro que hayamos visto, aunque no podamos decir qué es exactamente. Y no solo eso, sino que, posiblemente, una vez nos digan de qué animal se trata, en cuanto veamos una imagen de nuevo de dicho animal, vamos a saber reconocerlo inmediatamente. Es decir, una vez aprendido ese concepto, somos capaces de identificarlo cada vez que se nos presente. Y lo más normal es que para aprender nos baste con que nos hayan hablado de ese concepto una sola vez, o unas pocas veces como mucho.

Si trasladamos esta idea al campo tecnológico, una de las primeras preguntas que se nos puede venir a la mente es la siguiente: ¿podemos hacer que un ordenador aprenda por semejanza como lo hacemos los humanos? Más concretamente, si un dispositivo recibe un dato de entrada, de cualquier tipo, ¿es posible hacer que sea capaz de identificar de qué se trata basándose en la similitud con otros datos que ya haya recibido antes?

En primer lugar, tenemos que plantear cómo tratar los datos de entrada. Estos datos pueden ser prácticamente cualquier cosa que nos podamos imaginar, según el problema que se quiera tratar: imágenes, texto, medidas, colores, encuestas, etc. Es necesario extraer la información relevante de estos datos de forma que pueda ser tratada por un ordenador. En general, de cualquier tipo de dato vamos a poder extraer una serie de características que puedan ser representadas mediante un vector de números. Entonces, una vez disponemos de estos datos numéricos, ¿podemos establecer una medida de similitud entre ellos?

La respuesta es sí. Una vez tenemos nuestros vectores o puntos en el plano, el espacio, o cualquier dimensión superior, podemos medir distancias entre ellos, y en consecuencia, decir que dos ejemplos son similares si están cerca, y diferentes en caso contrario. Entonces,

los datos que forman la “memoria” de nuestro dispositivo contendrán todas las experiencias previas aprendidas, y cuando se recibe un nuevo dato, podemos observar su similitud con los datos memorizados, y en base a ello determinar qué tipo de dato es.

La Figura 3 muestra un ejemplo de este mecanismo de aprendizaje. Imaginemos que nuestros datos son imágenes de animales, que hemos conseguido transformar en datos numéricos bidimensionales, como los de la imagen, y queremos que nuestra máquina aprenda a reconocer, cada vez que nos llegue una nueva imagen de un animal, de qué animal se trata. Ya almacenados en la memoria de nuestro dispositivo tenemos los puntos coloreados, que representan a distintos tipos de animales: perro (azul), gato (rojo) y caballo (verde). Y ahora hemos recibido tres nuevas imágenes, (1), (2) y (3), que queremos averiguar a qué tipo de animal representan. El vector asociado a la imagen (1) está rodeado de puntos azules, lo que quiere decir que es parecido a los datos que ya teníamos almacenados de tipo ”perro”, y bastante diferente de los otros dos tipos, ya que se encuentra lejos de ellos. En consecuencia, nuestra máquina puede concluir que el ejemplo (1) es un perro. El ejemplo (2) está cerca de los puntos rojos, pero también está cerca de uno de los puntos azules. Posiblemente por la forma o los colores de la imagen, nuestro dispositivo encuentra similitudes tanto con las imágenes de gatos que teníamos almacenadas, como con las de perros. En cualquier caso, el vector asociado al ejemplo (2) se encuentra algo más cerca de los puntos rojos, por lo que finalmente el ejemplo (2) sería clasificado como gato. Por último, el ejemplo (3) está aislado de todos los demás. Esta situación sería la análoga a la de cuando nos enseñan una imagen de un animal que no hemos visto nunca. No es parecido a nada que hayamos visto con anterioridad, y por tanto, no podemos decir de qué animal se trata.

El ejemplo anterior aparenta emular bien el mecanismo de aprendizaje por semejanza humano, pero presenta dos inconvenientes. En primer lugar, los datos que aparecen representados en el espacio de la Figura 3 no sabemos cómo han sido obtenidos a partir de las imágenes. Aunque asumamos que los datos representan la información más relevante de las imágenes, no tenemos garantías de que al quedar representados en el plano, las imágenes más similares tengan que estar más cerca. O por lo menos, cerca en el sentido que estamos asumiendo, lo que va ligado al segundo inconveniente: y es que, en la Figura 3 hemos dado por hecho que el concepto de “cercanía” viene dado por la distancia usual en el plano, la *distancia euclídea*. Pero, ¿y si esta distancia no fuera la adecuada siempre? Fijémonos en la Figura 4. En este caso, vemos que los datos están dispuestos de forma extraña, pero se observa un patrón claro. La información relevante parece quedar representada en la coordenada vertical, la cual determina de qué tipo van a ser los ejemplos dados (A, B o C). Pero en este caso, los ejemplos nuevos que nos llegan (marcados con ’?’), son mal clasificados porque están más cerca de ejemplos con otra coordenada vertical, y la distancia euclídea no puede extraer dicha información. En general, vamos a disponer de infinitas formas de medir distancias, y lo más probable es que la distancia euclídea no sea la más adecuada en muchos casos. Este es el punto divergente con respecto al aprendizaje humano. Mientras que nosotros tenemos interiorizada una medida de similitud prácticamente infalible para cualquier par de objetos que se nos presenten, la búsqueda de una medida de similitud adecuada para un problema concreto es uno de los principales desafíos del aprendizaje por semejanza a

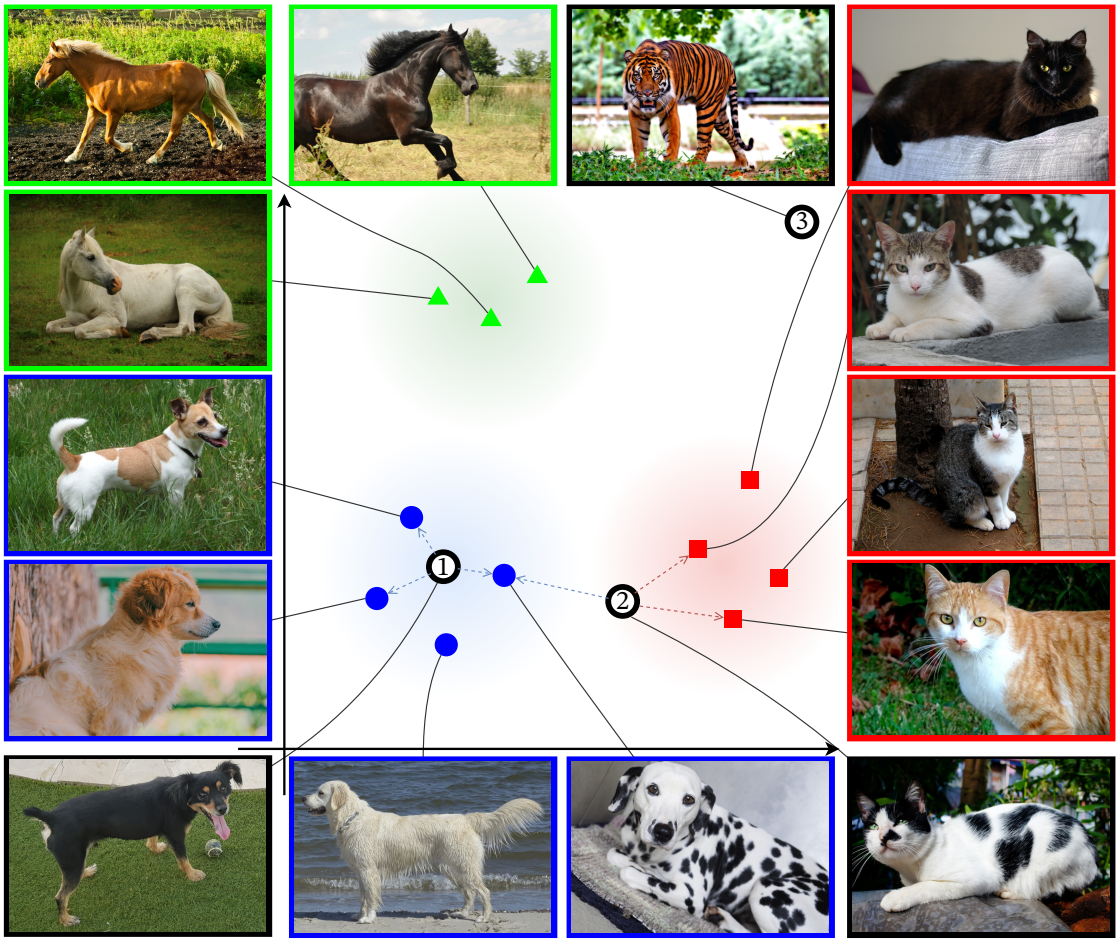


Figura 3: Aprendizaje por semejanza de tipos de animales.

nivel de máquina.

Al igual que en la Figura 3 usábamos los datos memorizados para tratar de identificar a los nuevos datos, puede resultar natural hacerse la siguiente pregunta: ¿y si también se puede encontrar una distancia adecuada aprendiéndola de esos mismos datos? Puesto que, idealmente, los datos del mismo tipo deberían ser cercanos y los datos de distinto tipo deberían estar alejados, lo que se conoce como *principio de uniformidad*, ¿por qué no extraer de los propios datos qué tiene que ser "cerca" y qué "lejos"? Esta es la base del *aprendizaje de métricas de distancia*, que formalizaremos más adelante. La finalidad de esta tesis doctoral en el área de investigación de la ciencia de datos es la profundización en esta disciplina, con el desarrollo de modelos que permitan resolver diferentes problemas en el ámbito del aprendizaje automático y la minería de datos.

En el panorama actual de la era digital, la cantidad de datos generados, recopilados y almacenados a nivel global ha experimentado un crecimiento exponencial. Este vasto océano

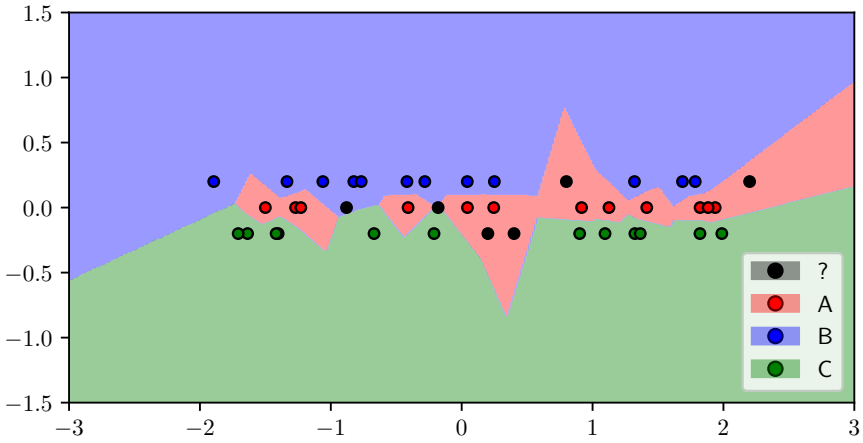


Figura 4: Cuando la distancia euclídea no es la adecuada.

de información alberga valiosos conocimientos y oportunidades que pueden transformar industrias, mejorar la toma de decisiones y ofrecer avances significativos en diversos campos. Sin embargo, la gestión, análisis y extracción de información útil de esta abrumadora cantidad de datos plantea desafíos significativos. Es aquí donde entra en juego el proceso de descubrimiento de conocimiento en bases de datos (*Knowledge Discovery in Databases* - KDD) [PF91], que busca convertir datos en información valiosa y, en última instancia, en conocimiento útil.

El KDD es un área interdisciplinaria que combina técnicas de minería de datos, aprendizaje automático, estadística y bases de datos para descubrir patrones, tendencias y relaciones ocultas en los datos. El proceso de KDD se puede dividir en varias etapas [MR05]: especificación del problema, extracción y selección de datos, preprocesamiento y transformación, minería de datos, interpretación y evaluación. El aprendizaje automático (*Machine Learning* - ML) [SSBD14] es la rama de la inteligencia artificial que se ocupa del diseño y desarrollo de algoritmos y modelos que permitan a los ordenadores la detección de patrones relevantes en los datos, y es de gran importancia en muchas etapas del KDD. Especialmente, en la minería de datos [HK06, A$^+$15], encargada de extraer el conocimiento de los datos, con la ayuda de las técnicas de ML, pero también en la etapa de preprocesamiento y transformación [GLH15], donde dichas técnicas pueden emplearse para detectar ruido, rellenar datos incompletos, seleccionar características relevantes y, en general, transformar los datos para que sean de mayor calidad [LGGRG$^+$20] de cara al proceso de minería de datos.

Dentro de las técnicas de ML, el *aprendizaje por semejanza* se basa en la idea de que datos de un mismo tipo tienen que ser similares, en el sentido de que su representación en el espacio en el que se trabaje tiene que ser parecida, de acuerdo con alguna medida de similitud o distancia que se defina. Este tipo de algoritmos pueden aplicarse en multitud de problemas de aprendizaje de interés general. Por ejemplo, dado un conjunto de datos etiquetados, podemos tratar de predecir las etiquetas de nuevos datos que nos lleguen en

base a la similitud con los datos que ya tenemos. O bien, dado un conjunto de datos sin etiquetar, podemos tratar de encontrar agrupamientos adecuados de datos de un mismo tipo en base a la similitud entre ellos. Las técnicas de aprendizaje por semejanza son conocidas desde los inicios del aprendizaje automático. Uno de los algoritmos más conocidos es el clasificador de los *k vecinos más cercanos* (*k*-NN - *k-nearest neighbors*) [CH67], el cual, dado un conjunto de datos etiquetados y un ejemplo para etiquetar, asigna a dicho ejemplo la etiqueta mayoritaria entre los *k* ejemplos más cercanos en los datos etiquetados, siguiendo la idea que se mostró en las Figuras 3 y 4.

Para que las técnicas de aprendizaje por semejanza como el *k*-NN funcionen, es necesario establecer una medida de similitud o, equivalentemente, una medida de distancia, entre los datos. Normalmente, se suelen utilizar distancias estándar, como por ejemplo, la distancia euclídea, en el caso de trabajar con datos numéricos. Sin embargo, estas distancias estándar pueden no ser las más apropiadas para los datos con los que trabajemos, ya que no tienen en cuenta las características subyacentes del conjunto de datos. El *aprendizaje de métricas de distancia* (DML - *Distance Metric Learning*) [XJRN03] surge con la finalidad de aprender las distancias a partir de los propios datos, de forma que puedan ser utilizados posteriormente por algoritmos de aprendizaje por semejanza con mejores resultados. En consecuencia, puede considerarse como un conjunto de técnicas de preprocesamiento y transformación de los datos para mejorar su calidad de cara a un aprendizaje por semejanza posterior.

En el aprendizaje automático, se distinguen tres grandes paradigmas: supervisado, no supervisado y semi-supervisado. El DML junto con el aprendizaje por semejanza engloban gran variedad de técnicas de diferentes naturalezas, teniendo alcance en los tres paradigmas:

- **Aprendizaje supervisado:** el objetivo de este paradigma es, dado un conjunto de datos etiquetados, aprender una función que permita predecir la etiqueta de nuevos datos. Según la naturaleza de las etiquetas, se distinguen dos problemas principales: la *clasificación* [DH$^+$06], en el que las etiquetas toman un conjunto finito de valores, y la *regresión* [DS98], en el que las etiquetas toman valores continuos. El DML puede utilizarse en el aprendizaje supervisado para acercar datos de una misma clase o con etiquetas cercanas, y alejar aquellos de clases diferentes. Esto permite mejorar el rendimiento posterior de algoritmos de clasificación o regresión por semejanza, como el *k*-NN.

- **Aprendizaje no supervisado:** en este paradigma, no se dispone de datos etiquetados, y el objetivo es encontrar patrones o estructuras en los datos. Este paradigma lo componen problemas como el *clustering* [Mir12], que consiste en encontrar agrupamientos de datos que sean similares. De nuevo, la similitud entre los datos vuelve a ser fundamental, y en consecuencia algoritmos de aprendizaje por semejanza como el *k*-means [M$^+$67] son de gran utilidad en este problema. El DML puede ser de gran utilidad en este problema si se dispone de información adicional sobre los datos, como veremos en el siguiente párrafo. Por otra parte, el DML es de gran utilidad en otro problema no supervisado: la *reducción de dimensionalidad*. Este problema consiste en reducir el número de variables en los datos de forma que se mantenga la mayor

parte de la información relevante. Tiene como objetivos la limpieza de los datos y la reducción del coste computacional de los algoritmos de aprendizaje, entre otros. La disposición de los datos en el espacio y las distancias entre ellos son fundamentales para encontrar una proyección adecuada de los datos, por lo que las técnicas de DML pueden utilizarse para encontrar tal proyección apoyándose en las distancias.

- **Aprendizaje por refuerzo:** la finalidad de este paradigma es aprender una política de actuación que maximice a largo plazo el beneficio obtenido por un agente en un entorno determinado [KLM96]. El agente interacciona con el entorno realizando acciones y el entorno responde con un beneficio y un nuevo estado. El DML puede utilizarse en este paradigma para aprender métricas de distancia que permitan codificar representaciones significativas de los estados y crear un espacio de atributos en los que los estados más similares sean cercanos [TKS11].

Los algoritmos de aprendizaje de distancias han sido ampliamente utilizados para reforzar algoritmos de aprendizaje por semejanza en problemas tradicionales como la clasificación, la regresión o el clustering [WS09, GHRS05, MVPC13, XJRN03, WT07]. El primer objetivo de esta tesis plantea el estudio de estos algoritmos y su replicación a nivel de software, con la finalidad de profundizar en el conocimiento de esta disciplina y de disponer de un framework que permita su aplicación experimental.

Sin embargo, el desarrollo tecnológico de los últimos años plantea nuevos desafíos, bien por la aparición de nuevos problemas de aprendizaje para los que es necesario adaptar las técnicas tradicionales, o bien por las nuevas herramientas de las que disponemos para abordarlos. En cuanto a los nuevos problemas que surgen, en esta tesis destacamos varias variantes del problema supervisado:

- **Aprendizaje semi-supervisado:** engloba problemas en los que se dispone de supervisión parcial [VEH20]. Esto puede traducirse de distintas formas. Por un lado, podemos disponer de dos subconjuntos de datos, uno etiquetado y otro no. El objetivo es aprovechar los datos no etiquetados, que suelen ser muchos más al ser menos costoso obtenerlos, para mejorar los resultados de clasificación o regresión. La propagación de la información de los datos etiquetados a los no etiquetados puede hacerse apoyándose en la similitud de los datos, por lo que de nuevo el DML entra en juego en este problema [ZG02]. Por otro lado, podemos disponer de *supervisión débil*, que consiste en que, aunque no conozcamos las etiquetas de los datos, sabemos que hay datos que tienen que ser del mismo tipo y datos que tienen que ser diferentes. El DML puede aplicarse para acercar los datos que tienen que ser similares y alejar los que tienen que ser distintos. Esto puede aplicarse sobre problemas como el *clustering con restricciones* [GAPDP+23], en el que se pretende encontrar agrupamientos adecuados sabiendo que hay ejemplos que deben estar en un mismo cluster y otros que deben estar en clusters diferentes.
- **Clasificación desbalanceada:** en este problema de clasificación, el número de ejemplos de cada clase es muy desigual, de forma que suele haber una o varias clases ma-

yoritarias y una o varias clases minoritarias que están poco representadas [FGG+18]. Esto a menudo provoca el sesgo de los algoritmos de aprendizaje hacia la clase mayoritaria, obteniendo un rendimiento pobre en las clases minoritarias. Al igual que ciertas técnicas basadas en semejanza han sido utilizadas para abordar este problema [Har68], la utilización simultánea de estas técnicas y DML puede ser de gran utilidad para conseguir mejoras sustanciales en el problema.

- **Clasificación ordinal:** en este problema de clasificación, las etiquetas de los datos toman valores de un conjunto ordenado [GPOSM+16], como por ejemplo, las estrellas de una valoración de un producto, o los niveles de gravedad de una enfermedad. En este caso, la similitud entre las etiquetas es importante, ya que cuanto más cercanos sean dos valores en las etiquetas, más parecidos deberían ser los datos. Integrar este comportamiento en una técnica de DML puede mejorar significativamente el rendimiento de los clasificadores por semejanza ordinales.
- **Clasificación monotónica:** es un problema de clasificación ordinal en el que, además, los datos de entrada también tienen restricciones de orden [CGK+19]. Por ejemplo, en un problema de predecir precios de casas, si todos los atributos de una casa son superiores a los de otra en un mismo barrio, es lógico que el precio de la primera sea mayor que el de la segunda. Al igual que en la clasificación ordinal, el DML puede ser de utilidad, pero en este caso, la medida de distancia debe tener en cuenta las restricciones de orden de los datos.

Estos problemas son de interés en numerosas áreas de aplicación real, como medicina [FDA23, HLW+16], industria [YWH+23], economía [LPC+23, CL14, ZG22] o detección de anomalías [TLL22]. El segundo objetivo de esta tesis se centra en proponer modelos para este tipo de problemas y la evaluación de su rendimiento.

Por otra parte, entre las nuevas herramientas que han surgido del desarrollo tecnológico de los últimos años, hay que destacar el *aprendizaje profundo* of *deep learning* [GBC17]. El deep learning ha revolucionado el campo del aprendizaje automático en los últimos años. Se basa en el uso de redes neuronales profundas, que son capaces de trabajar con tipos de datos muy variados y aprender representaciones de los datos de forma automática. Estas representaciones suelen ser de gran calidad, y en consecuencia, las redes neuronales profundas han conseguido resultados sobresalientes en multitud de problemas de aprendizaje automático. Sin embargo, el aprendizaje profundo también presenta nuevos desafíos, entre los que se destacan:

- **La necesidad de grandes cantidades de datos para aprender:** las redes neuronales profundas disponen de millones de parámetros para aprender, y por ello necesitan grandes cantidades de datos para ajustarlos. El *few-shot learning* (FSL) [WYKN20] es una disciplina que intenta abordar este problema. En los problemas de FSL, se dispone de un conjunto de datos de entrenamiento muy pequeño, y el objetivo es aprender a clasificar nuevos datos con esos pocos ejemplos, y posiblemente, un conjunto más grande auxiliar, de naturaleza diferente al que se desea aprender.

- **La falta de transparencia de los modelos:** las redes neuronales profundas son modelos muy complejos, y por ello, es difícil entender cómo funcionan y por qué toman las decisiones que toman. Esto es un problema en muchas aplicaciones, como por ejemplo, en medicina, donde es necesario entender por qué un modelo ha tomado una decisión para poder confiar en él. El desarrollo de una inteligencia artificial explicable (XAI - *explainable artificial intelligence*) es un campo de investigación emergente que trata de abordar este problema [ADRDS+20], no solo en las redes neuronales profundas, sino en cualquier modelo de aprendizaje automático en general. Las XAIs forman parte de un concepto más general denominado inteligencia artificial confiable. La inteligencia artificial confiable engloba un conjunto más amplio de principios y prácticas para asegurar que los sistemas de inteligencia artificial sean robustos, legales y éticos [AAES+23]. Uno de los requisitos para lograr este objetivo es la transparencia de los modelos, definida como la habilidad de un modelo para ser entendido por el ser humano de manera natural. Y un componente fundamental para la transparencia en un modelo es que sea capaz de dar cuenta de las decisiones que toma [DRDSC+23].

En los últimos años, se han propuesto modelos de DML que utilizan internamente redes neuronales profundas, dando lugar a un nuevo paradigma de aprendizaje: el *aprendizaje de distancias profundo* o *deep metric learning* [KB19]. El aprendizaje de distancias profundo busca combinar el potencial de aprendizaje de las redes neuronales profundas con el principio de uniformidad en el que se inspira el DML, para aprender modelos que sean capaces de representar los datos de forma óptima desde el punto de vista de la semejanza, y con un mayor rendimiento que los modelos clásicos de DML. Los modelos del aprendizaje de distancias profundo se diferencian de los de aprendizaje profundo en que los segundos aprenden a reconocer las distintas clases de datos, mientras que los primeros aprenden a discriminar entre ejemplos de distinto tipo sin la necesidad de reconocer explícitamente cada clase de las que dispone el problema, lo que ha demostrado ser de gran utilidad en problemas como el few-shot learning [LYMX23]. El desarrollo de modelos de aprendizaje de distancias profundo, y su aplicación en problemas complejos del aprendizaje profundo, como el FSL, constituye el tercer objetivo de esta tesis.

Por último, dos objetivos más se plantean en esta tesis. En primer lugar, como ya se ha mencionado, el desarrollo de modelos de aprendizaje que sean explicables es de interés en numerosas aplicaciones reales. El cuarto objetivo de esta tesis aborda el estudio de la explicabilidad en los modelos de aprendizaje de distancias y el aprendizaje por semejanza posterior, y la interacción entre ambos. El último objetivo de la tesis plantea la aplicación de los modelos desarrollados en problemas reales.

Para concluir esta introducción, presentamos un resumen de la estructura de esta tesis. La tesis se compone de tres capítulos: la disertación doctoral, en el Capítulo I, las publicaciones que avalan los conocimientos y conclusiones expuestos en la misma, en el Capítulo II, y el trabajo en progreso que completa los objetivos restantes propuestos en esta disertación, en el Capítulo III. La disertación la forman esta introducción y las siguientes secciones. La sección 2 presenta los fundamentos teóricos y conceptos utilizados a lo largo de la tesis. Las secciones 3, 4 y 5 presentan la justificación, los objetivos y la metodología sobre las que se

sustenta la tesis, respectivamente. En la sección 6 se presenta un resumen de la investigación llevada a cabo, y en la sección 7 se discuten los resultados obtenidos. Por último, la sección 8 presenta las conclusiones de la tesis y las líneas futuras de investigación.

El Capítulo II presenta las publicaciones que avalan los conocimientos y conclusiones expuestos en la disertación. Se presentan cuatro publicaciones: tres de ellas están publicadas en revistas indexadas internacionales y una de ellas está actualmente en revisión, en segunda vuelta. Las publicaciones son las siguientes:

- `pyDML`: A Python Library for Distance Metric Learning.
- A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges.
- Ordinal regression with explainable distance metric learning based on ordered sequences.
- Metric learning for monotonic regression: turning the space up to the limits of monotonicity.

Por último, el Capítulo III presenta el estado actual del último trabajo planeado para esta tesis, aún en progreso. Este trabajo está redactado también como un paper científico y su contenido y estructura son detallados en dicho capítulo.

2 Preliminaries

This section introduces the concepts required to understand the remainder of Chapter I. First, we present the notation and the foundations of the concepts discussed in this dissertation in Section 2.1. Then, the distance metric learning problem is introduced in Section 2.2. The supervised singular problems that have been discussed in this thesis are presented afterwards, in Section 2.3. Next, the modern challenges of machine learning and deep learning tackled in this thesis are introduced in Section 2.4. Finally, deep metric learning is discussed in Section 2.5.

2.1 Notation and preliminaries

This dissertation will mainly focus on the supervised learning paradigm. Unless otherwise specified, we will assume a set of N training samples $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i \in \mathcal{X}$, for $i = 1, \dots, N$ are the input samples and $y_i \in \mathcal{Y}$, for $i = 1, \dots, N$ are the corresponding labels. The input space $\mathcal{X}$ will depend on the nature of the data we are working with. When the input data is composed of d-dimensional numerical feature vectors, we will consider $\mathcal{X} \subset \mathbb{R}^d$, the d-dimensional Euclidean space. The output space $\mathcal{Y}$ will depend on the supervised learning task we are dealing with. The goal of supervised learning is to learn a function $f : \mathcal{X} \to \mathcal{Y}$ that maps input samples to their corresponding labels.

When working on Euclidean spaces, the use of matrices may be needed. We will denote as $\mathcal{M}_{d' \times d}(\mathbb{R})$ the set of all the matrices of dimension $d' \times d$. Recall that a matrix $L \in \mathcal{M}_{d' \times d}(\mathbb{R})$ can be biunivocally identified with a linear map $L \colon \mathbb{R}^d \to \mathbb{R}^{d'}$. Consequently, L can be thought of interchangeably as a matrix or as a linear map. We will also denote as $\mathcal{M}_d(\mathbb{R})$ the set of all the square matrices of order d. The definite matrices will be specially relevant in the development of this dissertation; we will denote as $\mathcal{M}_d(\mathbb{R})^+$ (resp. $\mathcal{M}_d(\mathbb{R})_0^+$) the set of all the positive definite (resp. positive semidefinite) matrices of order d. A matrix $M \in \mathcal{M}_d(\mathbb{R})$ is positive definite (resp. positive semidefinite) if and only if $\mathbf{x}^T M \mathbf{x} > 0$ (resp. $\mathbf{x}^T M \mathbf{x} \geq 0$) for all $\mathbf{x} \in \mathbb{R}^d \setminus \{\mathbf{0}\}$.

2.2 Background on distance metric learning

Distance metric learning (DML) is an ML task that consists in learning distances from a dataset. The learned distances can then be used to improve the performance of a wide range of ML algorithms. For any given non-empty set $\mathcal{X}$, a *distance* (also called *distance metric* or *metric*) over $\mathcal{X}$ is any function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with the following properties:

1. *Coincidence:* $d(\mathbf{x}, \mathbf{y}) = 0 \iff \mathbf{x} = \mathbf{y}$, for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$.

2. *Symmetry:* $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$, for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$.

3. *Triangle inequality:* $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y})$, for all $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{X}$.

A *pseudodistance* is a function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that, instead of the *coincidence* property, satisfies the weaker property $d(\mathbf{x}, \mathbf{x}) = 0$. Pseudodistances are also considered useful in metric learning-related tasks, and implicitly provide additional benefits such as dimensionality reduction. Following the standards in the literature, in this dissertation we will use the term *distance* to refer to both actual distances and pseudodistances.

Distances meet the minimum requirements for a consistent measurement, in the sense that removing any of the conditions in the definition results in scenarios in where the measurement for a pair of elements and the relative measurements between different pairs vary depending on the order in which they are taken. However, it is not clear whether the human similarity learning process follows the same rules [AP88]. That is why other similarity approaches that relax the symmetry and triangle inequality assumptions, such as *divergences*, have been proposed in the literature [BHS15]. In any case, the research on this topic is mostly focused on distances, and the scope of this dissertation is limited to the study of distances as well.

In the supervised setting, the ultimate goal of DML is to minimize a loss function $\ell(d, S, D, R)$, for $d \in \mathcal{D}$, where $\mathcal{D}$ is the search space containing the distances to use in the given problem. The sets S and D represent the similar and dissimilar pairs, respectively, which can be obtained directly from the labels if tackling a classification problem. The set R represents a remarkable feature in several algorithms and problems: triplet constraints (x_i, x_j, x_l), which impose that x_i should be more similar to x_j than to x_l. ℓ must be defined in a way that, when the loss is minimized, the similar samples are brought closer together and the dissimilar samples are pushed further apart. Each DML algorithm proposal approaches the definition of ℓ differently [WS09, GHRS05], and the choice of the loss function is a key decision to make when designing a new DML algorithm.

2.2.1 Linear distance metric learning

Working with continuous data in Euclidean spaces is a traditional approach, since any other data type can usually be transformed into numerical data using the right tools [GLH15, LLSD20], and the availability of distances for optimization is higher than in discrete spaces. In this scenario, the most common approach is to learn a Mahalanobis distance. For any $M \in \mathcal{M}_d(\mathbb{R})_0^+$, the Mahalanobis distance induced by M is defined as $d_M(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T M (\mathbf{x} - \mathbf{y})}$, for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$. Since every $M \in \mathcal{M}_d(\mathbb{R})_0^+$ can be decomposed as $M = LL^T$ with $L \in \mathcal{M}_d(\mathbb{R})$, the Mahalanobis distance can be equivalently expressed as $d_M(\mathbf{x}, \mathbf{y}) = \|L(\mathbf{x} - \mathbf{y})\|_2$, for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, i.e., the Euclidean distance after transforming the data with L. Therefore, the problem of learning a Mahalanobis distance can be reduced to the problem of learning a linear transformation $L \in \mathcal{M}_d(\mathbb{R})$. The choice of using M or L is often a matter of convenience and depends on the algorithm. Using M frequently leads to convex opti-

mization problems and optimality guarantees [XJRN03, DKJ+07], while L usually allows for dimensionality reduction, faster computations and data cleaning [WZ07, NMDB17].

2.2.2 Non-linear distance metric learning

The Mahalanobis approach is limited by the linearity of the transformation. Although this issue is not as noticeable as with linear classifiers, since the learned distance can be used later with non-linear classifiers such as the k-NN, a great learning potential is still being lost when the data distribution is complex. There are different ways of approaching non-linear distance metric learning. The most prominent are:

- **Kernel distance metric learning:** kernel DML approximations map the data to a higher dimensional space, in which the data ideally becomes linearly separable. Then, a distance metric is learned in the new space [TL07, NMDB17, WZ07]. The kernel trick is used to avoid the explicit computation of the mapping, which is computationally expensive, similarly to how it is applied in other learners such as Support Vector Machines (SVMs) [Bur98].
- **Local distance metric learning:** local DML approximations simultaneously learn several distances locally, in multiple regions of the space, like several clusters [WS09] or neighborhoods of each sample [WKW12]. This approach is specially useful when the data distribution is heterogeneous and the global structure is not well defined.
- **Deep distance metric learning:** deep metric learning has become a very active research area in recent years. It will be discussed in detail in Section 2.5.

2.3 Singular supervised learning problems

The supervised learning paradigm is the most common in ML. The standard classification and regression problems have been widely explored in ML in general and in DML in particular [WS09, GHRS05, MVPC13, WT07]. However, more and more problems of interest arise whose singularities render classic supervised learning-based approaches largely inadequate. In this section, we will introduce and describe the singular problems that have been addressed in this thesis. These problems are variants of the conventional classification problem, and are:

- **Imbalanced classification:** in imbalanced datasets there are significant differences between the number of samples representing each class [FGG+18]. This is a common problem in real-world applications, like disease diagnosis [AFB23] or fraud detection [KSVK23], where the positive (disease or fraud) samples are usually much less frequent than the negative (healthy or non-fraud) ones. The conventional classification algorithms are usually biased towards the majority classes, and thus the minority

classes are usually misclassified. The most common approaches to tackle this problem more effectively involve biasing the algorithms by giving greater weight to the minority classes [LLW02], or modifying the data distribution by oversampling the minority classes [CBHK02] or undersampling the majority classes [Har68].

- **Ordinal regression:** ordinal regression is an intermediate problem between classification and regression, in which the labels are ordered but not numerical [GPOSM+16]. This problem is common in applications like product rating [ZLL+17] or disease severity prediction [YTM+21], where there is a natural order among the labels (*very bad* < *bad* < *indifferent* < *good* < *very good*, or *severe* < *moderate* < *mild*, respectively), but the distances among them are not quantifiable. The algorithms for ordinal regression problems should take into account the varying importance of prediction errors: predicted labels farther away from the real ones should be penalized more [CK05, CK07].
- **Monotonic classification:** monotonic classification is a constrained ordinal regression problem in which, in addition to the order relation among the labels, the input data can also be ordered and the prediction should be monotonically increasing with respect to the input data [CGK+19]. Formally, if we denote the order relation in $\mathcal{X}$ as $\preccurlyeq$, and the order relation in $\mathcal{Y}$ as $\leq$, a monotonic problem satisfies that, for any given pair of samples $(\mathbf{x}_i, y_i)$ and $(\mathbf{x}_j, y_j)$, if $\mathbf{x}_i \preccurlyeq \mathbf{x}_j$ then $y_i \leq y_j$. This problem arises in areas like credit default prediction or medical diagnosis as well [SAM96], where the input data may be comparable (e.g. salary and debts, or age and cholesterol level, respectively) and the outputs (e.g. credit risk or disease severity, respectively) should be monotonically increasing with respect to the input data. Monotonic classifiers should extend the ordinal ones in order to be able to handle monotonicity constraints [DF08, CL14].

2.4 Modern challenges in machine learning and deep learning

Advances in artificial intelligence and machine learning research in recent years, and in particular the development of deep learning, have raised new challenges that need to be addressed. In particular, this thesis deals with two of them, already discussed in Section 1: the need for large amounts of data to learn in deep learning models, and the lack of transparency in certain ML models, including the deep learning ones. In this section, we present the problems that have been proposed to address these two challenges.

2.4.1 Few-shot learning

FSL [WYKN20] is a subfield of ML that addresses the challenge of training models to recognize and generalize from a limited amount of data. In traditional ML, and in deep learning in particular, models often require large datasets for effective training. However, in many real-world scenarios, collecting extensive data for every new task or category is impractical or cost-prohibitive. FSL aims to enable machines to learn quickly and accurately from just a

few examples—typically a handful or even a single example—per category. FSL is also useful in a scenario where data are imbalanced and present rare cases that are difficult to collect. And, ultimately, it lays the groundwork for moving towards human intelligence, capable of learning new concepts from a few examples.

The general FSL problem for classification is formulated as follows. Suppose we are given for training two different datasets, $\mathbb{D}_{\text{base}}$ and $\mathbb{D}_{\text{novel}}$. $\mathbb{D}_{\text{base}} = \{(\tilde{\mathbf{x}}_i, \tilde{y}_i) : \tilde{\mathbf{x}}_i \in \mathcal{X}_{\text{base}}, y_i \in \mathcal{Y}_{\text{base}}\}_{i=1}^{N_{\text{base}}}$ is the base dataset, which is a big auxiliary dataset used to make the model learn a general representation of the data, and does not need to be strictly related to the true data available for the problem. These true data are contained in $\mathbb{D}_{\text{novel}} = \{(\mathbf{x}_i, y_i) : \mathbf{x}_i \in \mathcal{X}_{\text{novel}}, y_i \in \mathcal{Y}_{\text{novel}}\}_{i=1}^{N_{\text{novel}}}$. The labels to predict are always different to the labels from the base set, i.e., $\mathcal{Y}_{\text{base}} \cap \mathcal{Y}_{\text{novel}} = \varnothing$. In the usual setup, both datasets are available and $N_{\text{base}} \gg N_{\text{novel}}$.

When testing, we are also given a pair (at least one, but we may have multiple pairs) of datasets, from the same domain as $\mathbb{D}_{\text{novel}}$: the *support set*, $\mathbb{D}_S = \{(\mathbf{x}_i, y_i) : \mathbf{x}_i \in \mathcal{X}_{\text{novel}}, y_i \in \mathcal{Y}_{\text{novel}}\}_{i=1}^{N_S}$, and the *query set*, $\mathbb{D}_Q = \{\mathbf{x}_1, \dots, \mathbf{x}_Q\} \subset \mathcal{X}_{\text{novel}}$, not necessarily labeled. The *few-shot learning classification* task consists in learning a classifier $f : \mathcal{X}_{\text{novel}} \to \mathcal{Y}_{\text{novel}}$ from $\mathbb{D}_S$ to predict the labels of the samples in $\mathbb{D}_Q$, assuming that N_S is very small (usually $N_S \leq 5$).

The support set is usually considered to have a few classes with the same number of samples for each class. When $\mathbb{D}_S$ contains C classes and N samples for each class, the problem is called *C-way N-shot* classification. In particular, when $N = 1$, the problem is called *one-shot* classification. A special case is the so-called *zero-shot* classification, in which the labels in the query set have not been seen in the support set (or just the support set is empty). The zero-shot classification problem is usually addressed by using auxiliary data, such as semantic information about the classes that were not present during training, in order to be able to predict them. The *generalized zero-shot* classification task assumes that the samples in $\mathbb{D}_Q$ can be both known and unknown. In this case, the task can also be to predict the class of a known sample and to detect the unknown samples. An FSL problem is said to be *cross-domain* when X_{base} and X_{novel} have different probability distributions (i.e. they are from different domains), and *cross-modal* when X_{base} and X_{novel} are of different natures (e.g. images and texts) [LYK+23].

Currently, multiple ways have been proposed to approach the FSL problem: data-based approaches focus on augmenting the datasets $\mathbb{D}_{\text{novel}}$ and $\mathbb{D}_S$ (and, possibly, $\mathbb{D}_{\text{base}}$) using prior information, to obtain a dataset big enough to make reliable predictions [QBL18, SBB+16]; algorithm-based approaches try to learn an initial learning parameter or a meta-learner using $\mathbb{D}_{\text{base}}$, to serve as a good starting point for the learning process with $\mathbb{D}_{\text{novel}}$ [CMPT+17, RL16]; and model-based approaches use the prior information from $\mathbb{D}_{\text{base}}$ to reduce the hypothesis space [KZS+15, VBL+16, SSZ17]. The deep metric learning methods that we will discuss in Section 2.5 combine features from these three approaches: the sampling strategies contribute to augment the dataset, the models are usually first trained on $\mathbb{D}_{\text{base}}$ to obtain a better starting point and transfer the knowledge to $\mathbb{D}_{\text{novel}}$ and, mainly, most of the models generate embeddings in a lower dimensional space that reduces the size of the hypothesis space.

2.4.2 Explainable artificial intelligence

An *explainable artificial intelligence* (XAI) can be defined as "*one that produces details or reasons to make its functioning clear or easy to understand*" [ADRDS+20]. The need for XAI has emerged due to the substantial advancements in ML model performance in recent years, which have not been accompanied by corresponding progress in model interpretability. This lack of interpretability is a major obstacle to the adoption of ML across domains like healthcare, finance, and law, where model decisions must be justified and explained. Moreover, the absence of interpretability can give rise to biased or unjust models, resulting in discrimination against specific demographic groups. Consequently, the development of XAI models is imperative to ensure that ML model decisions are equitable and impartial, fostering trust and confidence among users.

Concerning similarity-based learning and DML, which is the scope of this dissertation, it is interesting to analyze the explainability potential of these algorithms. The k-NN algorithm and, in general, any distance-based classifiers that make their predictions based on instances or prototypes can be considered transparent, since a human can naturally understand how the model internally works. In particular, as it was stated in Section 1, similarity-based learning algorithms are possibly the ones that match the human learning process the most. Transparency is another key concept, and it can be broken down into three levels [ADRDS+20]: simulatability, decomposability and algorithmic transparency. The k-NN classifier and similar methods can be easily simulated by a human for a reasonable k. Each part of the model can be understood separately, although a high number of variables or a complex distance measure can be obstacles for decomposability. Moreover, the lazy learning feature of these methods make them completely algorithmically transparent: it is really easy to understand why these models have made a decision by looking at the neighbors or prototypes of the predicted sample. The overall transparency of these classifiers is high, which makes them suitable for applications where the interpretability is a must.

The DML methods themselves are not so easy to interpret. The linear methods can be analyzed, since the metric matrix shows the interactions between each pair of samples. However, this is not enough to make a DML algorithm transparent. For non-linear methods, the interpretability is even more difficult. They can be considered black-boxes, specially when talking about the deep models. However, since they are usually designed and applied in conjunction with a similarity-based classifier, the aggregate model can still be considered a transparent model: the reason why a model makes a decision can still be understood by looking at the neighbors or prototypes. In addition, those neighbors or prototypes may be more informative due to the performance boost provided by the DML algorithm. To conclude, it is interesting to remark that k-NN can be considered a Case-Based Reasoning (CBR) system [AP94, LSG+19]. CBR systems encompass approaches to solving a problem that are inspired by other similar situations that have already been solved. Therefore, CBR is transparent in the same sense as k-NN. CBR is used as an explainable post-analysis for less transparent models. In the case at hand, the combination of DML and k-NN can be understood as a CBR system that arises organically because of the nature of both algorithms.

2.5 Deep distance metric learning

Deep distance metric learning or *deep metric learning* [KB19] is a subfield of both deep learning and DML that focuses on training neural networks to learn embeddings of data points in a way that preserves the similarity or dissimilarity relationships between them. The primary goal of deep metric learning is to map input data points into a high-dimensional space such that semantically similar data points are mapped closer to each other while dissimilar points are pushed farther apart.

Deep metric learning methods have enjoyed a great success in recent years. The powerful representation learning capabilities of deep neural networks allow deep metric learning methods to capture complex relationships in the data that traditional DML methods may not be able to identify. And, compared to other deep learning methods, deep metric learning focuses on learning embeddings that capture the fine-grained similarity structure of data, and their models tend to produce more discriminative representations; this makes them well-suited for tasks such as image retrieval, face recognition, and object tracking [WZW+17]. Deep metric learning models have also shown to be able to generalize and recognize previously unseen data points given only a few training samples [LYMX23].

Typically, deep metric learning models start from a base architecture capable of receiving input data and producing feature vector embeddings as output; the difference with other deep learning models lies in the way they are trained. The main general models in this discipline are detailed below.

Autoencoders. Autoencoders have become popular unsupervised deep learning models that encode the input data into a feature space and then reconstruct the input from the embedding [BKG23]. They are trained to minimize a reconstruction error loss function, which forces the model to learn embeddings that capture the most important features of the inputs. Autoencoders, as a generalization of the *principal component analysis* algorithm [Jol02]—which is considered an unsupervised linear DML algorithm for dimensionality reduction—can be considered unsupervised deep metric learning models. They are composed of two subnetwork modules, the encoder and the decoder, which are responsible for mapping the input samples to the feature space and for bringing them back to the original space, respectively.

Siamese networks and contrastive loss. Siamese networks [KZS+15] are one of the first neural network models designed for deep metric learning. This model contains two identical subnetworks that share the same parameters. When training, the network is fed two samples, one for each subnetwork. If both are expected to be similar (e.g. the samples belong to the same class), the network is trained to produce similar embeddings for both samples. They are considered a *positive pair*. If the samples are expected to be dissimilar (e.g. they belong to different classes), the network is trained to produce dissimilar embeddings. This pair of samples is called a *negative pair*. This training is achieved via a contrastive loss function that penalizes the network when the embeddings of positive pairs are too far apart, or when the embeddings of negative pairs are too close. The base contrastive loss function

is defined, for the pair of samples $(\mathbf{x}_i, \mathbf{x}_j)$, as:

$$L_{\text{contrastive}}(\mathbf{x}_i, \mathbf{x}_j) = (1 - y_{ij})\frac{1}{2}d(\mathbf{x}_i, \mathbf{x}_j)^2 + y_{ij}\frac{1}{2}\left[\max(0, m - d(\mathbf{x}_i, \mathbf{x}_j))\right]^2,$$

where $y_{ij} = 1$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar, and $y_{ij} = 0$ if they are dissimilar. The parameter m is a margin that controls the minimum distance between dissimilar samples, and d is the base distance to use in the embedding space (e.g. the Euclidean distance).

Triplet networks. Triplet networks [HA15] are an extension of siamese networks. They consider three identical subnetworks that share the same parameters. The network is fed a triplet $(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-) \in \mathcal{X}^3$, where x is the anchor sample, x^+ is a *positive* sample expected to be similar to x, and x^- is a negative sample expected to be dissimilar to x. The network is trained to move the anchor sample close to the positive sample, and far from the negative sample. The base triplet loss function is defined, for the triplet $(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-)$, as:

$$L_{\text{triplet}}(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-) = \max(0, m + d(\mathbf{x}, \mathbf{x}^-) - d(\mathbf{x}, \mathbf{x}^+)),$$

where m is, again, a margin constant. In general, the siamese and triplet networks can be extended to consider k-tuplets of any finite size [YT19]. The base loss functions can also be adapted to include regularization terms or other activation functions, such as the *softmax* or *softplus* functions, to normalize the outputs or soften the hard cut of the hinge loss of the base functions [ZLZ+21].

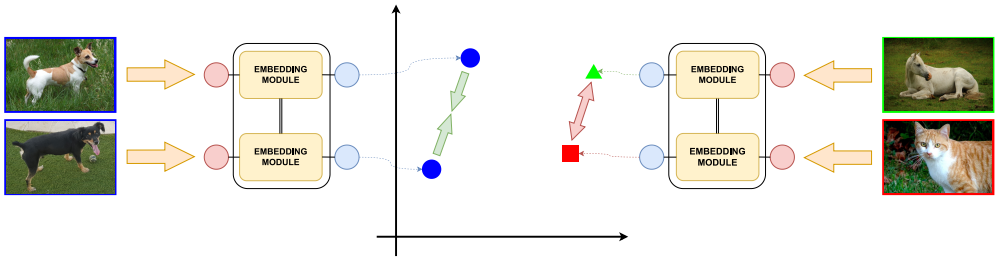
Matching networks. Matching networks [VBL+16] are a deep metric learning model designed to work on FSL problems. They also introduce *episodic training*, which is a training method that simulates the test phase and, therefore, makes the training phase more similar to a real-world scenario where few training samples are available. At each episode, the model is fed a unique support set (different from the one used in testing but with the same size and number of classes) and a query sample, randomly extracted from $\mathbb{D}_{\text{novel}}$. Through an attention mechanism, matching networks weigh the importance of the support samples for the query. Then, at the prediction stage, the network is fed both the support set and a test sample, and through the attention mechanism the most relevant training samples are selected to predict the label of the test sample, following a k-NN-like attention-weighted voting scheme.

Prototypical networks. Prototypical networks [SSZ17] are deep learning models that learn a representation for each class in the training or support data. During the training phase, following the episodic approach of matching networks, the model is fed a support set and a query sample. The embeddings of the support set are used to obtain a prototype for each class by averaging the embeddings of the samples belonging to each class. The loss function is computed as the log-softmax probability of the true class for the query point with respect to the prototypes. After each episode, the network is updated, modifying the embeddings and the prototypes of the given data. At the prediction stage, the class label for the test sample is obtained as the label of the prototype that is closest to the embedding of the test sample.

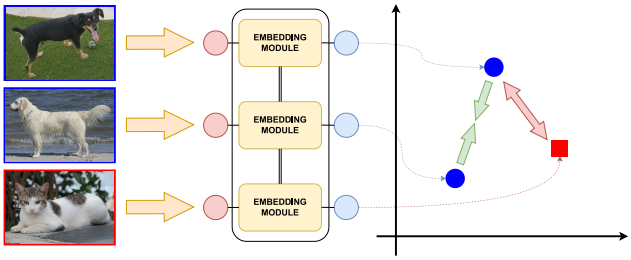
Relation networks. Relation networks are the first attempt at a deep metric learning model that directly learns a similarity function instead of computing embeddings or prototypes using discriminative loss functions [SYZ+18]. Following episodic training as well, both support and query samples are fed to an embedding module that produces a feature vector for each sample. Then, each support embedding is concatenated with each query embedding, and the concatenated vectors are fed to a relation module that computes a similarity score for each pair of samples. A mean squared error loss between the similarity scores and the ground truth is used to optimize the parameters of both modules. At the prediction stage, for each query sample, the label of the support sample with the highest similarity score is assigned to it.

Graph neural networks. Graph Neural Networks (GNNs) are deep learning models designed to work with graph data. They have become an interesting approach for deep metric learning, due to their ability to model the similarity scores in the data [SE18]. The general approach also consists in obtaining feature vectors through an embedding module. These embeddings will be the nodes to be fed to the GNN. Throughout the GNN layers, the edge values will be assigned, which will determine the final similarity between the data in the output layer. The labels will be assigned again according to the most similar support samples. For the training phase, a softmax layer is appended to transform the similarity scores into class probabilities, and a categorical cross entropy loss is used to update the model weights.

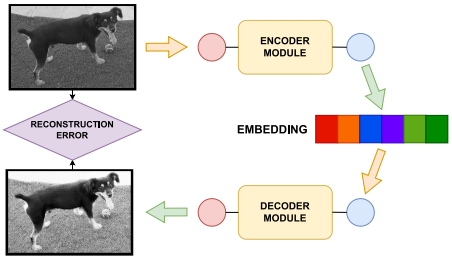
All the meta-architectures described above are illustrated in Figure 5. From these general models many other specific models can be derived, according to the specific needs of the problem to be solved [LYMX23, KB19, GGKC22]. Notice that, remarkably, there are other relevant factors to take into account when designing a deep metric learning model apart from the meta-architectures. The inner architecture modules used is one of these factors. The choice of loss function is also relevant, since the base loss functions described can be polished to adapt them to the problem and the type of data [PGH+20]. And, at last, a particularly important aspect in deep metric learning is the training and sampling strategy. The episodic training introduced in [VBL+16] has been shown to be a good approach for few shot learning problems, and the sampling strategy is determinant in models such as the siamese or triplet networks. A *hard mining* strategy, which focuses on taking the closest samples from different classes or the farthest samples from the same class, allows the model to learn faster and accelerate convergence. On the other hand, an *easy mining* strategy, which focuses on taking the closest samples from the same class or the farthest samples from different classes, can help the data within the same class to have a more compact representation and, therefore, be easier to classify [XSP20].



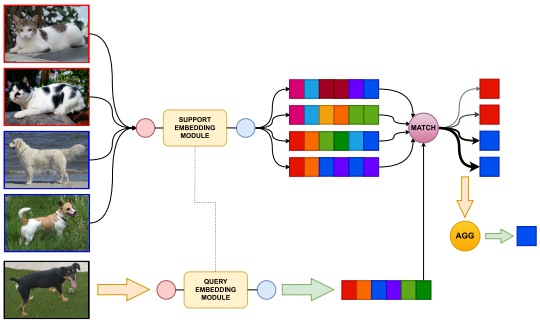
(a) Siamese networks.



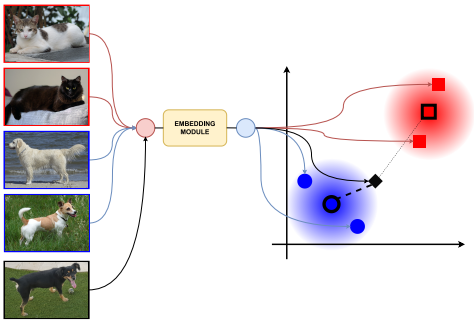
(b) Triplet networks.



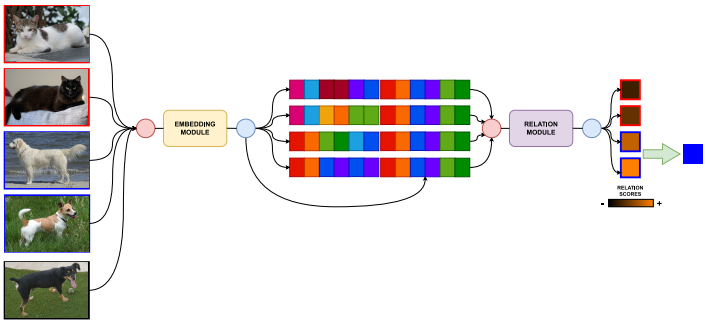
(c) Autoencoders.



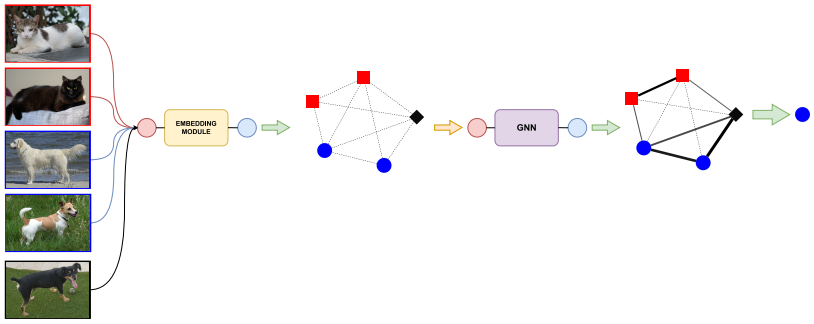
(d) Matching networks.



(e) Prototypical networks.



(f) Relation networks.



(g) Graph neural networks.

Figure 5: General architectures of the main deep metric learning models.

3 Justification

Within the realm of ML, similarity-based learning has evolved into a significant and noteworthy research area. Its intuitiveness and transparency makes it useful in applications where understanding the decisions made by the algorithms is crucial. The models are also often able to learn without requiring excessive amounts of data, and the simplicity of their designs means that they can be adapted to different types of problems.

DML has gained considerable relevance because of its connection to similarity-based learning algorithms and the enormous capacity of the former to improve the performance of the latter. DML algorithms can also be designed for a multitude of problems, as can similarity learners. They also have numerous learning paradigms which can be adapted to different types of data depending on their complexity. In particular, deep metric learning methods have grown along with the rise of deep learning and have demonstrated their potential to solve problems where other deep learning methods have more difficulty. The specific reasons that motivate this thesis on DML are listed below.

- Firstly, DML is a relevant area of research that was still in development at the beginning of this thesis project. The software tools available for DML were dispersed and incomplete and, although several studies on DML had been published, there was not a clear starting point for researchers interested in the area. Both these issues must be addressed in order to advance DML and to open it up to the scientific community.
- Secondly, most DML algorithms focus on standard classification problems, without taking into account the various peculiarities that many machine learning problems can present. The proposed thesis project attempts to exploit the potential of DML in unconventional variants of the classification problem, as the concept of what can be considered similar or not can be refined depending on the data being dealt with.
- Thirdly, deep metric learning research is at a key moment right now. The potential of deep learning has been demonstrated in many areas, and the aggregation with DML is no exception. The project proposes the application of deep metric learning to complex ML problems, such as FSL, which is a promising line of research where deep metric learning is expected to have a significant impact.
- Fourthly, the similarity-based learners are considered explainable models and can even be used to provide CBR explanations for less transparent models. But there is still a gap on how DML can impact the explainability of similarity-based models and how explainable the DML models themselves can be, and this thesis project aims to start filling this gap by analyzing the explainability of the DML models proposed.

To summarize, the proposed thesis project constitutes a substantiated contribution to the advancement of DML. This contribution is justified by the significance of the field and the demand for a comprehensive reference that encompasses both software and literature. It also addresses the limitations of existing unconventional classification techniques and explores the potential of applying deep metric learning to complex problems. Furthermore, it seeks to examine the intricate relationships between similarity-based learning, DML, and the aspect of explainability.

4 Objectives

After presenting the fundamental concepts from the state-of-the-art, we can delve into the objectives that motivate this thesis. The first one is to provide a general reference and a unified software framework for the DML area. Then, with that foundation already established, the next objectives tackle the development of new DML models, including shallow models in singular classification problems, and deep models for complex ML problems. Finally, the last objectives revolve around the explainability of the DML models proposed and the application of DML to real problems. These objectives are detailed below.

Creation of a unified software framework and a general reference for DML. The first objective of this thesis is to provide a general reference for the field of DML. This reference will be a tutorial consisting of a review of the current state of DML—including an introduction to DML and its mathematical foundations—, a taxonomy of the most important existing models, and an experimental analysis of their performance. This reference will be complemented with a software framework that will allow the implementation of the most representative DML models. This framework will be designed to be extensible, so that new models can be easily added to it, and it will also include a set of tools to facilitate the analysis of the models implemented.

Development or adaptation of DML algorithms for singular problems. As it has already been observed, DML algorithms are mostly designed for standard classification problems. However, there are many other problems with singularities that make them a poor match for conventional algorithms. Among these singular problems, several non-standard variants of the classification problem stand out: imbalanced classification, ordinal regression and monotonic classification. The development of models to address these problems has constituted the second objective of this thesis.

Deep metric learning and complex problems. Deep metric learning has grown to be a research topic with effective applications in many problems, such as FSL. The aim of this objective includes the implementation, study and visualization of the deep metric learning meta-architectures, the analysis of the complex problems in deep learning and how to approach them with deep metric learning, and the development of a model to address a complex problem.

Explainability analysis of the developed models. It is interesting to explore how the DML models interact with the explainable features of the similarity-based learners. In this way, there are different aspects to explore: how DML can improve the intuitiveness of the results provided by a similarity-based method, how well the decision made by a DML model with a similarity-based classifier can be understood by a human, and how the DML model itself can provide additional conclusions that may also be useful for the user.

Application of DML to real problems. Finally, the last objective of this thesis is to apply the developed models to real problems. The aim is to demonstrate the potential of DML in real applications, and to analyze the performance of the models in real scenarios.

5 Methodology

The development of this thesis has been carried out based on the following method of work and experimentation, inspired by the scientific method:

- **Observation.** Study of the state of the art of distance metric learning and its extensions to deep learning. Analysis of complex problems and their possible modelling by means of distance metric learning techniques.
- **Hypothesis formulation.** Design of distance metric learning and deep distance metric learning algorithms for singular and complex problems.
- **Collection of observations.** Application of the developed algorithms to different datasets belonging to the different problems dealt with and obtaining results.
- **Hypothesis testing.** Analysis of the quality of the proposals, comparing them with other state-of-the-art techniques.
- **Proof or refutation of hypotheses.** Acceptance, rejection or modification of techniques developed as a result of the tests carried out.
- **Thesis or scientific theory.** Extraction, drafting and acceptance of conclusions drawn during the process.

6 Summary

The knowledge amassed in this thesis is disseminated across five distinct studies, three of them already published in scientific journals, one submitted and under peer review, and one work in progress with the aim of publication. The purpose of this section is to provide a concise overview and introduction to these studies, with a detailed discussion of their results to follow in Section 7. The list of the works already published or submitted to review is presented below, while the work in progress is discussed in Chapter III:

- Suárez, J. L., García, S. & Herrera, F. (2020). `pyDML`: a Python library for distance metric learning. Journal of Machine Learning Research, 21(96), 1-7.
- Suárez, J. L., García, S. & Herrera, F. (2021). A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges. Neurocomputing, 425, 300-322. DOI: `https://doi.org/10.1016/j.neucom.2020.08.017`
- Suárez, J. L., García, S. & Herrera, F. (2021). Ordinal regression with explainable distance metric learning based on ordered sequences. Machine Learning, 110(10), 2729-2762. DOI: `https://doi.org/10.1007/s10994-021-06010-w`.
- Suárez, J. L., González-Almagro, G., García, S. & Herrera, F. (2023). Metric learning for monotonic classification: turning the space up to the limits of monotonicity.

The rest of this section is structured in accordance with the publications listed above and the goals detailed in Section 4. Firstly, Section 6.1 presents a summary of the software developed and the tutorial created. Secondly, Section 6.2 summarizes the studies related to the development of DML algorithms for singular problems. Afterwards, Section 6.3 introduces the studies related to the application of deep metric learning to complex problems. Finally, Section 6.4 shows a summary on how explainability has been analyzed in the developed proposals, while Section 6.5 outlines the applications of DML that have been carried out in real problems.

6.1 Creation of a unified software framework and a general reference for DML

At the beginning of this thesis project, shallow DML was a well-known area of research but it lacked two main components to increase accessibility to the ML community: a unified software framework to make DML readily available and easy to use for researchers, and a general reference that could serve as a starting point for researchers interested in the area.

The first component, the unified software framework, was developed in the first study associated with this thesis. The framework, called `pyDML`, is a Python library that implements the most representative DML algorithms. It is designed to be fully compatible with

the Scikit-Learn algorithms [PVG+11] and extensible, so that new algorithms can be easily added to it. It also includes a set of tools to facilitate the analysis of the models implemented, such as functions to plot and tune the models. The publication associated with this study is:

> Suárez, J. L., García, S. & Herrera, F. (2020). `pyDML`: a Python library for distance metric learning. Journal of Machine Learning Research, 21(96), 1-7.

The second component was developed as a tutorial on DML that was published in the second study associated with this thesis. This tutorial encompasses a study of distance metric learning that includes its mathematical foundations, an analysis of the most relevant DML algorithms in supervised, semi-supervised and unsupervised scenarios (including around 20 of them), an experimental analysis of the studied methods in 34 well-known datasets from the literature, a Bayesian statistical analysis of the results, and a discussion of the prospects and challenges of the area. The tutorial is complemented by an appendix that includes a comprehensive description of the algorithms analyzed and the mathematical foundations studied. The expectation is that this tutorial will serve as a general reference for the DML area, and as a starting point for researchers looking to enter this field. In conjunction with the developed software framework, this research effort was also expected to help us to learn in depth about DML methods and to create an extensive basis to start the development of the models foreseen in the following objectives. The publication associated with this study is:

> Suárez, J. L., García, S. & Herrera, F. (2021). A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges. Neurocomputing, 425, 300-322. DOI: `https://doi.org/10.1016/j.neucom.2020.08.017`

6.2 Development or adaptation of DML algorithms for singular problems

As it was mentioned above, most DML algorithms have been designed for standard classification problems and their potential in singular problems can still be further explored. To fulfill this objective, two different studies have been conducted. The first one tackles the development of a new DML algorithm for ordinal regression problems, while the second one explores possibilities to apply DML to monotonic classification problems.

6.2.1 DML for ordinal regression

One of the main issues of ordinal regression is how to tackle this problem properly from a purely ordinal perspective. It is common either to consider the problem as a classification problem by ignoring the ordinal nature of the data, or to consider it as a regression problem by assuming that we can quantify the differences between labels—which may not be the

case. DML is a promising approach in this context, since it can be applied to bring samples closer in the embedding space the more similar their labels are in the output space.

In the first study associated with this objective, we propose a new DML algorithm for ordinal regression problems that is built around the concept of ordered sequences. In the neighborhood of any sample in the training set, we define two concepts of *ordered sequence*: one in the input space, which consists in sequences of samples with increasing distances to the anchor sample, and another one in the output space, which consists in sequences of labels whose values go up or down with respect to the anchor label. When a sequence is ordered in both spaces, we call it a *chain*. The proposed algorithm, called Chain Maximizing Ordinal Metric Learning (CMOML), is trained to maximize the number of chains in the neighborhood of each sample.

Since the objective function of CMOML is not differentiable, a black-box differential evolution optimizer [Pri13] is proposed. A kernel version of the algorithm, Kernel Chain Maximizing Ordinal Metric Learning (KCMOML), is also proposed to deal with non-linear problems. The algorithms are tested on a wide variety of ordinal datasets and compared with other DML proposals for ordinal regression and the general state-of-the-art in this area. The publication associated with this study is:

> Suárez, J. L., García, S. & Herrera, F. (2021). Ordinal regression with explainable distance metric learning based on ordered sequences. Machine Learning, 110(10), 2729-2762. DOI: `https://doi.org/10.1007/s10994-021-06010-w`.

6.2.2 DML for monotonic classification

Monotonic classification restricts ordinal regression by adding monotonic constraints to the data. If two samples can be compared, then their labels should also be comparable, and with the same relation as in the input space. The difficulty that arises with DML in this problem compared with unconstrained ordinal regression is that a transformer-based approach (learning L instead of M) may easily violate the monotonic constraints. To learn a non-metric-based approach it is crucial for the transformed space to not be modified enough that monotonic constraints start being violated or variables stop being meaningful.

To do this, in the second study associated with this objective we propose a new DML algorithm for monotonic classification that learns a linear map to transform the input space without introducing new monotonic constraints that were not present in the original space. The proposed algorithm is called Large Margin Monotonic Metric Learning (LM^3L), and makes use of two special types of matrices: *monotonic matrices* and *M-matrices*. These matrices have very interesting properties that allow the monotonicity of the dataset to be preserved when they are used as a linear transformation. The algorithm is trained using a variation of the traditional large margin loss [WT07] adapted to the ordinality of the problem, and compared with the Euclidean distance using different distance-based classifiers, both monotonic and non-monotonic. The publication associated with this study is:

Suárez, J. L., González-Almagro, G., García, S. & Herrera, F. (2023). Metric learning for monotonic classification: turning the space up to the limits of monotonicity.

6.2.3 Additional work on DML for singular problems

In addition to the two proposed studies for ordinal regression and monotonic classification that make up the second objective and part of the publications of this thesis, additional work on DML for imbalanced classification has been carried out. This work combines the potential of Neighborhood Component Analysis (NCA) [GHRS05]—one of the most powerful DML algorithms for k-NN classification—and the *condensed nearest neighbors* rule [Har68]—a similarity-based undersampling technique that synergizes well with the NCA optimization mechanism. The hybridization of these two techniques results in a new DML classifier, called Condensed Neighborhood Component Analysis (CNCA), which is able to learn a distance and undersample an imbalanced dataset at the same time. This study is part of a conference paper and is not included in this thesis, but it will be mentioned in Section 8.2.

6.3 Deep metric learning and complex problems

Deep metric learning, as a part of deep learning, has experienced significant growth thanks to the boom of the deep learning field. Its greater discriminative capacity makes deep metric learning algorithms a very powerful complement to the rest of deep learning, with great potential in problems such as FSL, where deep learning struggles to perform effective learning.

In this thesis project, as part of an international internship at the University of Cardiff, a collaboration with the Knowledge Representation and Reasoning and the Natural Language Processing (NLP) groups of the University of Cardiff was carried out. The aim of this collaboration was to apply deep metric learning to the problem of FSL in the context of Relation Extraction (RE).

The *relation extraction* problem [BB07] is an NLP problem that focuses on identifying and extracting semantic relationships between entities mentioned in text. The goal is to determine the type of relationship that exists between two or more entities in a given sentence or document. This is typically achieved by analyzing the text to identify specific patterns, keywords, or linguistic cues that indicate the nature of the relationship.

As an example, let us consider the sentence “Madrid is the capital of Spain”. In this sentence, the entities are “Madrid” and “Spain”, and the relationship between these entities is “is_the_capital_of”. In this example, relation extraction involves identifying that “Madrid” and “Spain” are related, and that the relationship between them is “is_the_capital_of”.

The RE problem is found in many fields, playing a crucial role in areas such as: constructing knowledge graphs [ZFH23], which are useful in applications related to semantic web,

question answering or data integration; sentiment analysis [ZHX+22], where extracting relations between entities and their associated sentiments can yield a deeper understanding of opinion and sentiment in text; recommendation systems, since extracting relations between users, products, and their preferences can be used to provide users with more personalized and relevant recommendations [WZZ+23]; biomedical and scientific research [SC22], where relation extraction can be used to find relationships between genes, proteins, drugs, diseases, and other entities, for tasks such as drug discovery or disease modeling.

Few-shot relation extraction is a specialized RE task where the goal is to extract semantic relationships between entities in text with very limited training data. In traditional relation extraction, models are trained on a large dataset with labeled samples of entity pairs and their corresponding relations. However, few-shot relation extraction deals with scenarios where only a small number of training samples are available—often just a few or even one—from which to extract relations between entities. Few-shot RE has applications in a variety of domains where collecting labeled data is challenging, such as in the biomedical domain or other domains with many technical terms in general, where there is a lack of labeled data.

The study associated with this objective focuses on the application of deep metric learning to the few-shot RE problem in the FewRel dataset [GHZ+19], which is a popular benchmark dataset for few-shot RE. Current few-shot RE approaches focus on *contrastive pretraining* [PGH+20], which consists in pretraining an NLP model such as BERT [DCLT19] with a contrastive loss function (in a siamese meta-architecture), to bring pairs of sentences with the same relation closer in the embedding space and push pairs of sentences with different relations farther away. The pretraining is done on a large dataset extracted using Wikipedia as a corpus and Wikidata [VV19] as the knowledge graph. Then, the model is fine-tuned on the *FewRel* dataset using a prototypical network with episodic training. This base model has been extended in several ways. For example, by adding more learning tasks to the pretraining model [QLT+21] or by weighting the labels from the pretraining dataset depending on how noisy they are likely to be [HLS22].

In this study, we propose to extend the contrastive pretraining approach in several ways:

- Firstly, we add to the loss function an additional term consisting in an autosupervised contrastive loss. This loss is computed by taking positive pairs using a different strategy. For any sentence given, a partner sentence for the positive pair is made by randomly masking parts of the sentence that do not include the entities. These positive pairs are also fed to the model, and the loss is computed as the contrastive loss between the original sentence and the masked sentence. This is done to shift the focus of the model away from parts of the sentence that may be irrelevant to learn the relations.
- Secondly, we modify the mining strategy for the pairs that are sampled in the supervised contrastive loss term. We identify that some of the samples may be more reliable than others in terms of how many sentences with the same relations are present in the neighborhood of the samples, how many different pairs of entities appear in that neighborhood, and if the sentence structure varies sufficiently within the neighborhood. We make those *reliable samples* more likely to be fed to the model.

- Finally, we hypothesize that the embeddings generated by BERT are not optimally distributed in space for the RE task at hand. Samples from identical relations may be split into different clusters, which is not ideal. Therefore, we propose the application of additional metric learning on the embeddings obtained from the pre-training to facilitate further learning in the prototypical network.

This study is a work in progress. The preliminar work associated with this study is discussed in Chapter III.

6.4 Explainability analysis of the developed models

The need for explainable models has dramatically increased in recent years. Due to the explainable characteristics of similarity-based models, it is of interest to study explainability also in DML and, therefore, this study is proposed as an objective. Note that this objective is cross-sectional, since the study to be carried out is applicable to any DML model developed.

Specifically, in this thesis this objective has been addressed in the proposed model for ordinal regression discussed in Section 6.2.1. A similarity-based classifier like the k-NN used in the experiments can be seen as a CBR system, which gives us a point of view from which to analyze the explainability of the proposed models.

The proposed explainability study selects several datasets from the experimentation as a study case. On these datasets, the CBR-based explanations of the k-NN classifier with the Euclidean distance and the k-NN classifier with the distance learned by CMOML are analyzed. This is done by taking the 3 nearest neighbors of each test sample by each of the models. For the selected datasets, we make a displayable graphical representation of the samples and their neighbors. The task is to identify, from these representations, if there are features in the neighbors that are contributing effectively to the decision made by the model, and how these features change according to the model used.

As an additional task, we also explore how well CMOML can make bidimensional and tridimensional embeddings, so that under these spaces the decisions made by the model can be visualized and understood. This is done by learning a dimensionality reduction map $L \in \mathcal{M}_{d' \times d}(\mathbb{R})$ using the CMOML algorithm, and analyzing the outputs in the datasets selected for the explainability study.

6.5 Application of DML to real problems

In the practical section, the implementation of ML models in real problems is essential to gauge the scope and real applicability of these models. The successful application of a model in a real problem means that the developed model is empirically validated.

This objective is again cross-cutting, as applicability can be assessed in any developed model. In the context of this thesis, this objective has been addressed in the development of the deep metric learning model in Section 6.3. This model has been developed and applied to the few-shot relation extraction problem, as explained in that section.

7 Discussion of Results

Of the objectives set out in the thesis, the second and third objectives require experimental analysis to demonstrate the performance of the models developed. In the first objective, in addition, an experimental analysis of the most important methods in the classical DML literature is carried out. It is therefore important to follow a consistent experimental methodology that guarantees that the results are reliable and comparable, and that is complemented with a statistical validation that demonstrates that the results are robust, generalizable and not the result of chance.

The experiments developed to meet the objectives of this thesis follow this logic, using validation techniques for the training and prediction of models, specific metrics for each problem addressed that show the goodness of the models developed, comparison with leading algorithms in the problem and the families of algorithms selected for it, and Bayesian statistical tests [BCDZ17] that validate the comparisons established between the different algorithms.

This section summarises the results obtained in each of the studies that complete the objectives of this thesis. Analogous to Section 6, this section is structured in accordance with the publications listed above and the goals detailed in Section 4. Section 7.1 summarizes the conclusions drawn from the literature study of DML, the software development and the experimental analysis conducted. Section 7.2 shows the results obtained with CMOML and LM^{3}L in ordinal regression and monotonic classification, respectively. Section 7.3 presents the current state of the approach developed for the few-shot RE problem. The last objective, the application of DML to real problems, is also analyzed in this section, since it is part of the same study. Finally, Section 7.4 analyzes the conclusions extracted from the explainability analysis performed on CMOML.

7.1 Creation of a unified software framework and a general reference for DML

The two studies developed to cover this objective have resulted in contributions of great interest in different parts of the field of DML.

The software library developed in the first study, `pyDML`, provides an accessible implementation of the most popular methods of traditional DML, highlighting algorithms designed for supervised classification but also including unsupervised and weakly supervised ones. It currently exceeds 20 implemented algorithms, including implementations of models developed in this thesis. In addition to the algorithms, additional tools have been incorporated, such as a plotting module that allows for the visualization of the distances and the classification grid of the algorithms, and a tuning module that facilitates the hyperparameter selection for the algorithms.

`pyDML` features continuous integration, conforms to the Python PEP8 style guide and

maintains a changelog and standard documentation powered by Sphinx and Numpydoc. The algorithms in the library follow a hierarchical class structure that also adapts to the Scikit-Learn hierarchy, making it easily extensible. Furthermore, it is available on GitHub[1] under the GPL license and therefore accessible for community contribution through issues and pull requests. At the time of writing, the GitHub repository has 169 stars. `pyDML` is also available in the official Python package index, PyPI.

The second study, the tutorial on DML, offers an overview of shallow DML and its algorithms. In the theoretical study of DML and its algorithms, a novel classification for the algorithms is proposed, which is composed of: algorithms specialized in dimensionality reduction, algorithms specialized on improving distance-based classifiers, information theory-based algorithms, and kernel methods. This theoretical study helped to identify the main areas to take into account when developing new DML algorithms and to propose new perspectives and challenges for future directions in the field.

Finally, the experimental analysis performed in the tutorial has allowed us to compare the performance of the most representative DML algorithms not only in a standard k-NN classification setup, but also in several additional setups such as Nearest Class Mean (NCM) classification, kernel classification and dimensionality reduction. The experimental analysis was performed with the algorithms implemented in `pyDML`.

The conventional k-NN setup consisted in evaluating the accuracy of the k-NN classifier after applying each of the DML algorithms. This was done by using a stratified 10-fold cross validation on 34 well-known classification datasets from the literature. The datasets were min-max normalized prior to the execution of the algorithms to give the same importance to all the features at the time of computing the distance. The distances were evaluated with the k-NN classifier considering several values for k: 3, 5 and 7. Then, the results obtained were validated using Bayesian sign pairwise tests to determine whether the differences between each pair of algorithms were statistically significant. This main experiment has laid the foundations for the rest of the experiments carried out throughout the thesis. The cross-validation, normalization, classification and statistical validation schemes have been mostly kept intact in the rest of the experiments in the whole thesis project, with some necessary variations depending on the problem.

The remaining experimentation that composed this tutorial consisted in, firstly, a performance analysis of the algorithms specialized in NCM classification, which followed a similar scheme to the one used in the conventional k-NN setup, but replacing the k-NN with the NCM classifier. Then, the kernel based methods were also evaluated using again the k-NN classifier; however, several kernels for each algorithm were tried in order to identify which kernel was the most suitable for each algorithm. Finally, the dimensionality reduction experiments were made to show the potential of the DML algorithms as dimensionality reduction methods. In this case, the DML algorithms that are capable of performing dimensionality reduction were applied to map the data to different dimensions. Then, the k-NN classifier was applied again to evaluate the results on the different dimensions.

[1] `https://github.com/jlsuarezdiaz/pyDML`

The overall results of this extensive experimentation helped us to identify the most powerful algorithms in terms of k-NN classification and NCM classification, which was consistent with the algorithmic classification proposed in the study. This has been especially important when proposing new algorithms in the following objectives, since their designs have been heavily influenced by the best performing algorithms in some cases. It was also shown that the kernel versions of the algorithms were able to improve the results in the more complex datasets when the appropriate kernel was chosen. Finally, the capability of several DML algorithms to reduce the dimensionality of the data was also corroborated, being able to keep producing good results in low dimensions in many cases or even improving them.

7.2 Development or adaptation of DML algorithms for singular problems

The two studies related to this objective prove the feasibility of applying DML to singular problems. The problems tackled in both studies are proven to be modelable in a DML-like framework, since the label distributions in the space can be set in correspondence with the distances between the samples in the space. In particular, CMOML and LM^{3}L have been proven to outperform the distance-based state-of-the-art for ordinal regression and monotonic classification, respectively, and to even be competitive with the general state-of-the-art in these areas. Sections 7.2.1 and 7.2.2 discuss the results obtained with these two methods and the conclusions drawn from the results, respectively.

7.2.1 DML for ordinal regression

CMOML, our DML proposal for ordinal regression, has been tested on 36 different datasets; 23 of them are real-world ordinal datasets, while the remaining 13 are equal-frequency discretized datasets for regression. CMOML has been evaluated using a stratified 5-fold cross validation appended to a median-vote 7-NN classifier, which is the natural extension of the classic majority-vote k-NN for ordinal regression. The results obtained by CMOML have been compared with the results of the Euclidean distance and of the main DML proposal for ordinal regression [NMDB18], with the same classifier. The Concordance Index (C-Index) metric, which measures the ratio between the number of correctly ordered pairs and the total number of pairs, has been used to compare the results.

The results, validated by a Bayesian sign test, show that CMOML clearly outperforms both other distances. The experimental analysis was extended to include several main state-of-the-art ordinal regression methods: two variants of the SVMs for ordinal regression [CK07, LL12], an extended bayesian network for ordinal regression [HWL20] and a kernel extreme learning machine for ordinal regression [SLY$^+$19]. The same experimentation scheme was used, and the results obtained by CMOML remained competitive with the state-of-the-art.

7.2.2 DML for monotonic classification

Our DML proposal for monotonic classification, LM^{3}L, has been tested on 10 different well-known monotonic datasets from the literature. LM^{3}L is, to the best of our knowledge, the first DML proposal for monotonic classification. Therefore, the aim of the experimental analysis is to show whether the distance learned by LM^{3}L is able to outperform the Euclidean distance under different scenarios. For this purpose, we considered several variants of the k-NN classifier, both monotonic and non-monotonic: the traditional and the median-vote k-NNs, the monotonic k-NN variants [DF08] and the monotonic fuzzy k-NN variants [GGL$^+$21]. A stratified 5-fold cross validation was used to evaluate the results.

Three different metrics have been considered in this study. To measure the quality of the prediction performance, we chose two ordinal metrics: Mean Absolute Error (MAE) and C-Index. To measure the monotonicity of the predictions, we proposed two Non-Monotonic Index (NMI) variants to observe how many pairs remained monotonic in the training set after being transformed and how many test samples interfered with the monotonicity of the training set. We also reported the total number of comparable pairs, since after the data transformation some pairs may not be comparable anymore. The results, validated with Bayesian sign tests, show that LM^{3}L when combined with the median-vote k-NN classifier outperforms the Euclidean distance in the ordinal performance metrics, while also contributing to the reduction of the non-monotonicity of the dataset at the cost of losing some comparable pairs. It is remarkable how this combination is able to outperform the monotonic classifiers, despite the median-vote not being monotonic and LM^{3}L having also been tested with the monotonic classifiers.

7.3 Deep metric learning and complex problems

The *FewRel* dataset for the few-shot RE problem presents a private test set evaluated through an online platform to which the predictions are uploaded and from which the evaluation results are obtained. Consequently, the methodology followed for the evaluation of the model presented for this objective will be based on the training of the model with the public *FewRel* data, and its subsequent evaluation through the platform with the private data. The FewRel test set is composed of a multitude of problems of four types: 5-way 1-shot, 5-way 5-shot, 10-way 1-shot and 10-way 5-shot. For each problem, the support set and the query to be predicted with that support set are given. The final metric used is the average hit rate of the queries on each problem.

The FewRel test set is divided into three different subsets of different nature:

- **FewRel 1.0:** consists of data of the same nature as the base FewRel training set.
- **FewRel 2.0 Domain Adaptation (DA):** the data in this test set are obtained from biomedical databases, which are different in nature from the one that composes the FewRel training set. This is a much more challenging problem, since no similar data

has been seen during training and the model must acquire a sufficiently large discriminative capacity to be able to predict them correctly.

- **FewRel 2.0 none-of-the-above (NOTA):** this test set can be considered a zero-shot variant of the problem. In this case, it is not guaranteed that the relation to be predicted is present in the support set; consequently, it is possible for a model to predict *NOTA* to indicate such absence. This task is evaluated differently, under 5-way 1-shot and 5-way 5-shot classification considering two different NOTA rates in the labels: 15 % and 50 %.

The work under development for the problem has already demonstrated competitive results with respect to the base contrastive pretraining approaches currently proposed as a solution to the problem. Our current model has been tested in *FewRel 1.0* and *FewRel 2.0 DA*, and incorporates the autosupervised loss component and the mining strategy described in Section 6.3. Both of the components separately contribute to the improvement of the results, and the combination is outperforming the base contrastive pretraining in most of the problems. There is still work to be done in order to obtain more significant differences, for which the third component of the proposal described in Section 6.3, together with the refinement of the mining strategy and the autosupervised loss importance, are expected to be key.

7.4 Explainability analysis of the developed models

In addition to analyzing how CMOML outperforms the main DML proposal for ordinal regression and is competitive with the state-of-the-art, the explainability of CMOML has also been analyzed. Unlike the other compared methods from the state-of-the-art, CMOML with the k-NN classifier is able to provide explanations for the predictions it makes.

As a CBR system, CMOML with k-NN is able to explain the decision it makes in terms of the neighbors selected for the sample to test. In the analysis performed, where we compared the CBR results between CMOML and the Euclidean distance, we observed that the neighbors that CMOML obtains exhibit features that are more relevant to the decision made by the model than those present in the neighbors obtained by the Euclidean distance. In terms of human reasoning, this means that the neighbors obtained by CMOML are more likely to be the neighbors that a human would select. This makes CMOML a reliable explainable approach for ordinal regression when combined with k-NN, which makes it more useful than the compared methods in applications where it is crucial to understand the decisions made by the model.

8 Conclusions and Future Work

This section concludes the thesis (Section 8.1), gathers all the relevant studies we have published (Section 8.2), and provides notes on future research lines (Section 8.3).

8.1 Conclusions

This thesis proposes an innovative and rigorous analysis of DML, which provides both a global vision of the topic, at the theoretical and implementation level, and new perspectives, with the development of three new DML proposals in different areas. The general objective of the thesis is to broaden the knowledge on DML and approach it from new points of view. For this purpose, a bibliographic review was elaborated together with a software library to carry out an in-depth study of the discipline and exhaustive experimental work to demonstrate and validate the quality of the developed proposals.

To achieve the first objective, the most relevant algorithms in the field of DML were gathered and subjected to exhaustive analysis. This led to the development of a software library that integrates these algorithms in the same homogeneous environment. Complementing this library, a tutorial was developed which explains in detail both the basics of DML and the developed algorithms, and compares them experimentally. The development of the rest of the proposals of this thesis was aided by the insight gained from the analysis of the existing algorithms and the identification of their strengths and weaknesses.

The second objective concerns the development of DML algorithms in singular problems, and focused mainly on two problems: ordinal regression and monotonic classification. The first problem was addressed through the CMOML algorithm, which tries to unify the input and output spaces from a combinatorial point of view, considering sequences of data in the neighborhoods and a heuristic optimization. CMOML proved to be dominant compared to the main DML proposals for ordinal classification, and competitive with respect to the state-of-the-art in the discipline. The second problem was tackled through the LM^{3}L algorithm. This proposal addressed monotonic classification with DML for the first time, and takes advantage of the properties of a subset of matrices to respect the monotonicity of the data. LM^{3}L was shown to outperform distance-based classifiers with Euclidean distance, both monotonic and non-monotonic, when combined with the appropriate classifier.

The third objective focuses on a currently hot area: deep metric learning, which has grown in popularity with the rise of deep learning due to its ability to complement it in problems where classic deep learning models run into more difficulties. In particular, in this objective we have tackled the problem of FSL applied to relation extraction, a problem of great interest in NLP. For this purpose, relevant improvements were introduced into the main FSL proposals for RE, inspired by contrastive pre-training together with a prototypical network. These improvements, which consist in refining the loss function to take into account additional information, modifying the mining strategy to train with more reliable samples, and introducing additional DML to optimize the feature vectors obtained by the

model, have currently led to improvements with respect to the base model, and more significant improvements are expected to be achieved by further polishing these proposals.

Finally, the last two objectives involve the study of the explainability in the developed models and the application of the models to real problems, respectively. The first of these has been addressed on CMOML. The analysis of the explainability of this model has led to the conclusion that, when combined with the k-NN classifier, the two of them form a CBR system in which the neighbors considered in the model to make the decisions reveal more identifying features that resemble those that would influence actual human decision-making. As for the last objective, it is answered via the proposal for the few-shot relation extraction problem addressed with deep metric learning in the third objective.

Conclusiones

Esta tesis propone un análisis del DML desde un punto de vista innovador y riguroso, que proporciona tanto una visión global, a nivel teórico y de implementación, de la temática y las propuestas más relevantes en el área, como nuevas perspectivas, con el desarrollo de tres nuevas propuestas de DML en distintos ámbitos. La tesis tiene como objetivo general ampliar los conocimientos sobre el DML y abordarlo desde nuevas perspectivas. Para ello, se ha elaborado una revisión bibliográfica acompañada de una biblioteca software para realizar un estudio profundo de la disciplina y estudios experimentales exhaustivos para demostrar y validar la capacidad de las propuestas desarrolladas.

Para lograr el primer objetivo, se han recopilado los algoritmos más relevantes en el ámbito del DML, para los cuales se ha realizado un análisis exhaustivo. Esto ha concluido con la elaboración de una biblioteca software que integra a estos algoritmos en un mismo entorno homogéneo. Complementando a esta biblioteca, se ha desarrollado un tutorial que detalla tanto los fundamentos del DML como de los algoritmos desarrollados, y los compara experimentalmente. El análisis de los algoritmos ha permitido identificar las fortalezas y debilidades de los mismos, las cuales se han aprovechado para desarrollar el resto de propuestas de esta tesis.

El segundo objetivo aborda el desarrollo de algoritmos de DML en problemas singulares, y se ha centrado principalmente en dos problemas: la clasificación ordinal y monotónica. El primer problema se ha abordado mediante el algoritmo CMOML, que busca unificar los espacios de entrada y salida desde una perspectiva combinatoria, considerando sucesiones de datos en los vecindarios y una optimización heurística. CMOML ha demostrado ser dominante en comparación a las principales propuestas de DML para clasificación ordinal, y ser competitivo con respecto al estado del arte de la disciplina. El segundo problema se ha abordado con el algoritmo LM^{3}L. Esta propuesta aborda por primera vez la clasificación monotónica con DML, y aprovecha las propiedades de un subconjunto de matrices para respetar la monotonía de los datos. LM^{3}L ha demostrado ser capaz de mejorar a los clasificadores basados en distancias con la distancia euclídea, tanto monotónicos como no monotónicos, cuando se combina con el clasificador adecuado.

El tercer objetivo se centra en un área candente en la actualidad: el aprendizaje de distancias profundo, el cual ha crecido en popularidad junto con el auge del aprendizaje profundo, debido a su capacidad de complementarlo en problemas donde los modelos clásicos de aprendizaje profundo tienen más dificultades. En particular, en este objetivo se ha abordado el problema de FSL aplicado a la extracción de relaciones, un problema de gran interés en el procesamiento del lenguaje natural. Para ello, se han introducido mejoras relevantes en las principales propuestas de FSL para RE, inspiradas en pre-entrenamiento contrastivo junto con una red de prototipos. Estas mejoras, consistentes en el refinamiento de la función de pérdida para considerar información adicional, la modificación del muestreo de ejemplos para entrenar con ejemplos más fiables y la introducción de DML adicional para optimizar los vectores de atributos obtenidos por el modelo, han supuesto actualmente mejoras con respecto al modelo base, y se espera conseguir mejoras más significativas perfeccionando estas propuestas.

Por último, los dos últimos objetivos abordan el estudio de la explicabilidad en los modelos desarrollados y la aplicación de los modelos a problemas reales, respectivamente. El primero de ellos se ha abordado sobre CMOML. El análisis de la explicabilidad de este modelo ha permitido concluir que, cuando es combinado con el clasificador k-NN, forman conjuntamente un sistema CBR en el que los vecinos que se consideran en el modelo para tomar las decisiones dejan a la vista rasgos más identificativos que son más cercanos a los que podría considerar un humano para tomar la misma decisión. En cuanto al último objetivo, este se está abordando en la propuesta para el problema de FSL para extracción de relaciones abordado con técnicas de aprendizaje de distancias profundo en el tercer objetivo.

8.2 Publications

This section lists journal, conference and preprint papers published during the PhD study period, sorted by publication date. The DOI and the number of citations indicated by Google Scholar are given for journal and conference papers.

- **Journal papers**:

  1. Suárez, J. L., García, S. & Herrera, F. (2020). `pyDML`: a Python library for distance metric learning. Journal of Machine Learning Research, 21(96), 1-7. CITED BY: 21
  2. Suárez, J. L., García, S. & Herrera, F. (2021). A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges. Neurocomputing, 425, 300-322. DOI: `https://doi.org/10.1016/j.neucom.2020.08.017`. CITED BY: 153
  3. Suárez, J. L., García, S. & Herrera, F. (2021). Ordinal regression with explainable distance metric learning based on ordered sequences. Machine Learning, 110(10), 2729-2762. DOI: `https://doi.org/10.1007/s10994-021-06010-w`. CITED BY: 7

- **Under revision**:

  1. Suárez, J. L., González-Almagro, G., García, S. & Herrera, F. (2023). Metric learning for monotonic classification: turning the space up to the limits of monotonicity. Submitted to: Applied Inteligence.

- **Conference papers**:

  1. Suárez, J. L., García, S. & Herrera, F. (2021, September). Distance metric learning with prototype selection for imbalanced classification. In International Conference on Hybrid Artificial Intelligence Systems: 16th International Conference, HAIS 2021, Bilbao, Spain, September 22-24, 2021, Proceedings 16 (pp. 391-402). Cham: Springer International Publishing. DOI: `http://dx.doi.org/10.1007/978-3-030-86271-8_33`. CITED BY: 1
  2. Suárez, J. L., García, S. & Herrera, F. (2021, October). Ordinal regression with explainable distance metric learning based on ordered sequences: extended abstract. In 2021 IEEE 8th International Conference on Data Science and Advanced Analytics, DSAA 2021, Porto [Online], Portugal, October 6-9, 2021 (pp. 1-2). IEEE. DOI: `https://doi.org/10.1109/DSAA53316.2021.9564133`. CITED BY: 0.
  3. Suárez, J.L., González-Almagro, G., García, S. & Herrera, F. (2022, July). A preliminary approach for using metric learning in monotonic classification. In Advances and Trends in Artificial Intelligence. Theory and Practices in Artificial Intelligence: 35th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2022, Kitakyushu, Japan, July 19–22, 2022, Proceedings (pp. 773-784). Cham: Springer International Publishing. DOI: `https://doi.org/10.1007/978-3-031-08530-7_65`. CITED BY: 0

8.3 Future work

The results of this thesis open up new avenues for research in DML and suggest new challenges in the discipline that may be of interest, both in the field of ML and data mining and for the scientific community in general. This section presents these lines of work and challenges that have emerged throughout the development of this thesis.

- **DML and big datasets.** Although this problem has been mitigated with the rise of deep metric learning, shallow DML methods, which are mostly based on the computation of the distance matrix, are still not scalable to big datasets in terms of both number of samples and features. Feature selection, dimensionality reduction, big data and parallelization techniques provide a lot of possibilities to improve the efficiency of the most computationally expensive shallow DML methods.

- **DML and trustworthy AI.** The boom in AI technologies in recent years has made it necessary to consider how it should be regulated in order to avoid some of the major risks it may pose, such as the generation and spread of misinformation, bias that discriminate against certain groups, or the violation of data privacy. In this scenario, the concept of trustworthy AI arises to lay the foundations for the development of a responsible and safe AI. XAI, as a component of AI transparency, plays a key role in trustworthy AI. XAI has already been addressed in this thesis, but there are still open questions and lines of research in this area [AAES+23]. Some examples related to DML are the following:

  - The extension to deep metric learning models, to learn more powerful 2D or 3D embeddings that can be easily visualized and understood by humans.
  - The discriminative approach lacks the confidence measure that the recognition approach has. Transforming similarities into a reliable confidence measure is a challenge that remains unaddressed.
  - The development of methods that provide explanations at attribute level, highlighting which features in the input data may be most influential in determining the similarity between instances.
  - The development of DML models where real-time decision-making is essential, such as autonomous driving, where the model must be able to explain its decisions in real time.

  More generally, trustworthy AI also opens up new areas of research [DRDSC+23] where DML can be useful. Some to highlight are:

  - **Fairness.** Analyze the bias of a distance learned by a model to ensure that similarity metrics are not biased against certain groups [Cho17]. Future work may involve creating distance metrics that are sensitive to fairness constraints and minimizing bias in similarity assessments [EFS23].
  - **Data privacy.** Develop models that do not compromise the privacy of the data owners. DML may be useful in several privacy preserving techniques. For instance: in secure multi-party computation and homomorphic encryption [YPB+14], it may be useful to learn distances that preserve the relationships between the encrypted and decrypted spaces; in differential privacy [Dwo06], DML algorithms that are able to add controlled noise to the pairwise distances may be of interest [HWM+20]; in federated learning [ZXB+21], it must be taken into account that embeddings cannot be shared between different data owners—how to learn a distance for each client and how to aggregate the distances is the key factor to develop a federated DML model [PHY21].
  - **Robustness.** Develop models that are robust to adversarial attacks or data perturbations [AM18]. In particular, similarity-based methods may help to improve the robustness of deep models and to detect adversarial samples [MZY+19].

  - **Data quality.** Place a primary focus on the quality, accessibility and management of data. The quality of the data used to train ML models is critical to achieving high performance AI systems [ZBL+23]. Sometimes it can be even more important than the model itself. It is therefore necessary to implement techniques to assess the quality of the data, to preprocess it and to mitigate biases. When synthetic data is generated (e.g., in data augmentation or generative models), it is important to validate this process to ensure that the data is trustworthy. Distance-based algorithms, and hence DML algorithms, can play an important role in many aspects of data quality, such as outlier and corner case detection, or density analysis for synthetic data generation.

- **Multi-domain and multi-modal metric learning.** Most of the DML literature has a limited scope of learning a single distance function for a single dataset. However, with the great technological advances of recent years and their implementation in a wide variety of industries, this may fall short of addressing real-world problems. On the one hand, it is common to find problems that involve multiple datasets which may be of different domains. This can be due to multiple data sources, varying data collection times or locations, or an insufficient amount of data of interest that needs to be complemented by a bigger dataset from a different domain. Deep metric learning has already started to be used in multi-domain tasks, such as the FSL problem addressed in this thesis, but there is still a lot of work ahead in this area towards a more general and robust multi-domain metric learning.

  On the other hand, with the rise of social networks, multimedia platforms and areas such as computer vision or NLP, there is an increasing need for models that can simultaneously deal with data of different natures, such as images, text, videos, numerical data, etc. Multi-modal learning arises with the aim of generating models capable of tackling this task. With the development of deep learning and its ability to deal with any type of data, this problem has gained popularity [SLS+21]. The extension to deep metric learning is the natural next step. Merging all the data of different modalities into the same embedding space becomes key, and in turn how to train the model to discriminate adequately taking into account all modalities also becomes key.

- **Beyond distance metric learning: general similarity learning.** DML algorithms are a subgroup of the similarity learning algorithms that require the learned similarity functions to be distances. The analytical properties of distances such as the Euclidean, Malahanobis or kernel distances make them suitable for many optimization tasks and the distance properties leave no room for ambiguity about what is considered similar and what is not. DML has captured nearly all the attention within similarity learning since the early days of the discipline. However, when comparing to the human similarity-based reasoning, it is not clear that all the distance properties are relevant. As an example, consider the triangle inequality, which states that two items will always be more similar than the sum of the similarities of each item with an intermediate item; it is easy to find examples where this does not apply in human reasoning. For instance, a human may not consider a person and a horse similar at all, but they may

consider both similar to a centaur, thus violating the triangle inequality. Therefore, it is of interest to go deeper into more generic similarity metrics, going hand in hand with psychology, to pave the way towards a similarity learning closer to that of human beings.

- **Metric learning towards a general purpose AI.** *General purpose AI* represents the goal of creating AI systems that can perform a wide range of tasks with human-level proficiency [TMP+24]. While currently, AI systems tend to be specialized for specific tasks, the long-term vision is to develop AI models that can adapt and generalize its learning to new domains. Combining the insights discussed above, DML can play an important role in the development of general purpose AIs. A sufficiently generic similarity measure capable of effectively comparing cross-modal data may be a key factor in the development of AI models that can address diverse tasks with limited data, mirroring human-like problem-solving abilities. Trustworthy AI is essential for general purpose AI as well, as the broader the AI's capabilities, the more critical it becomes to ensure ethical, explainable and accountable behavior.

Chapter II

Publications

«*If a machine is expected to be infallible, it cannot also be intelligent.*».
– Alan Turing.

1 pyDML: A Python Library for Distance Metric Learning

- **Journal:** Journal of Machine Learning Research (JMLR)
- **JCR Impact Factor:** 3.654
- **Rank:** 48/139
- **Quartile:** Q2
- **Category:** Computer Science, Artificial Intelligence
- **Status:** Published

Ref.: Suárez, J. L., García, S. & Herrera, F. (2020). pyDML: a Python library for distance metric learning. Journal of Machine Learning Research, 21(96), 1-7.

pyDML: A Python Library for Distance Metric Learning

Juan Luis Suárez [*,a,b] **Salvador García** [a,b] **Francisco Herrera** [a,b]

[a] *Department of Computer Science and Artificial Intelligence (DECSAI), University of Granada, Granada, Spain*
[b] *Andalusian Institute of Data Science and Computational Intelligence (DaSCI)*

Abstract

pyDML is an open-source python library that provides a wide range of distance metric learning algorithms. Distance metric learning can be useful to improve similarity learning algorithms, such as the nearest neighbors classifier, and also has other applications, like dimensionality reduction. The pyDML package currently provides more than 20 algorithms, which can be categorized, according to their purpose, in: dimensionality reduction algorithms, algorithms to improve nearest neighbors or nearest centroids classifiers, information theory based algorithms or kernel based algorithms, among others. In addition, the library also provides some utilities for the visualization of classifier regions, parameter tuning and a stats website with the performance of the implemented algorithms. The package relies on the scipy ecosystem, it is fully compatible with scikit-learn, and is distributed under GPLv3 license. Source code and documentation can be found at https://github.com/jlsuarezdiaz/pyDML.

Keywords Distance Metric Learning · Classification · Mahalanobis Distance · Dimensionality · Python.

1 Introduction

The use of distances in machine learning has been present since its inception, since they provide a similarity measure between the data. Algorithms such as the nearest neighbor classifier [1] use that similarity measure to label new samples. Traditionally, standard distances, like the euclidean distance, have been used to measure the data similarity. However, a standard distance may not fit our data properly, so the learning results could be non op-

* Corresponding Author (jlsuarezdiaz@decsai.ugr.es)
Email addresses: salvagl@decsai.ugr.es (Salvador García), herrera@decsai.ugr.es (Francisco Herrera)

timal. Finding a distance that brings similar data as close as possible, while moving non similar data away can significantly increase the quality of similarity based learning algorithms. This is the task that distance metric learning carries out.

Distance metric learning (*DML*) [2, 3, 4] is a machine learning discipline with the purpose of learning distances from a dataset. If we focus on *Mahalanobis distances*, which are expressed as $d_M(x, y) = \sqrt{(x-y)^T M(x-y)}$, where M is a positive semidefinite matrix, learning a distance reduces to learning the matrix M. This is equivalent to learning a linear map L. In this case, $M = L^T L$ and the learned distance is equivalent to the euclidean distance after applying the transformation L to the data. Therefore, DML algorithms aim at optimizing functions parameterized by a positive semidefinite (also called metric) matrix M, or by a linear map L [5, 6].

Several DML libraries have been developed in different programming languages. In R, we can find the package `dml` [7], which proposes 11 DML algorithms. However, only 5 algorithms are currently implemented and the package has not exhibited activity for some time. In MATLAB, the `DistLearn`[1] toolkit provides links to several DML implementations, many of them corresponding to the original authors. However, many of the links are currently broken and the algorithms are not presented in a unified framework. In Python, the `metric-learn` library [8] provides 9 different DML algorithms, mostly oriented towards weak supervised learning, with the exception of a few classical supervised DML algorithms.

In this paper, we present the `pyDML` library, a Python package that gathers a wide variety of DML algorithms. The following sections describe the main features of the software, some instructions for installation and usage, several quality standards to which the project subscribes, and finally we expose our plans on future functionalities to be included in the project.

2 Software Description

The `pyDML` library currently contains around 20 (mostly supervised) algorithms and distances that can be used to prepare a dataset for a subsequent similarity-based learning. These algorithms, and some of their main features, are shown in Table 1.

Python, the chosen programming language, is widely used in machine learning, and has several libraries specialized in this field. The main one is `Scikit-Learn` [23], an efficient open-source library for machine learning, which relies on the Scipy[2] ecosystem, which contains numerical calculus libraries, such as `NumPy`, data processing libraries, such as `Pandas`, or data visualization libraries, such as `Matplotlib`. Python has been also chosen because until now it does not have an extensive library with supervised DML algorithms. `pyDML` tries to fill this gap, providing numerous supervised DML algorithms, both classic algorithms and new proposals.

[1] `https://www.cs.cmu.edu/~liuy/distlearn.htm`
[2] `https://www.scipy.org`

Algorithm/distance	Supervised	Dimensionality reduction	Oriented to improve...	Information theory based	Kernel version	Learns ... (L or M)
Euclidean	✗	✗	Just a static distance	✗	✗	Identity
Covariance	✗	✗	Just a static distance	✗	✗	M
PCA [9]	✗	✓	Non-specific	✗	✗	L
LDA [10]	✓	✓	Non-specific	✗	✗	L
LLDA [11]	✓	✓	Non-specific	✗	✗	L
ANMM [12]	✓	✓	k-NN	✗	✗	L
LMNN [5]	✓	✓	k-NN	✗	✗	Both
NCA [13]	✓	✓	1-NN	✗	✗	L
NCMML [14]	✓	✓	Nearest class mean	✗	✗	L
NCMC [14]	✓	✓	Generalized NCM	✗	✗	L
ITML [15]	Weak	✗	Non-specific	✓	✗	M
DMLMJ [16]	✓	✓	k-NN	✓	✗	L
MCML [17]	✓	✗	Non-specific	✓	✗	M
LSI / MMC [18]	Weak	✗	Non-specific	✗	✗	M
DML-eig [6]	Weak	✗	Non-specific	✗	✗	M
LDML [19]	✓	✗	Non-specific	✗	✗	M
GMML [20]	Weak	✗	Non-specific	✗	✗	M
KDA [21]	✓	✓	Non-specific	✗	✓	L
KLLDA [11]	✓	✓	Non-specific	✗	✓	L
KANMM [12]	✓	✓	k-NN	✗	✓	L
KLMNN [22]	✓	✓	k-NN	✗	✓	L
KDMLMJ [16]	✓	✓	k-NN	✓	✓	L

Table 1: Current algorithms/distances available in the pyDML library.

The design followed for the development of the algorithms has preserved the structure of the algorithms of the `Scikit-Learn` library. In particular, the DML algorithms are included in the group of transformation algorithms, where the transformation consists in applying the learned linear map to the samples. Therefore, the implemented algorithms inherit from a template class `DML_Algorithm`, which in turn inherits from the `sklearn.base.TransformerMixin`[3] class of the `Scikit-Learn` toolkit. This hierarchy allows the DML algorithms to be treated as black-box transformers, which facilitates their handling and *pipelining* with other `Scikit-Learn` algorithms. The `DML_Algorithm` class provides the inherited methods `fit(X,y)` and `transform(X)`, to learn the distance and apply it to the data, following the `Scikit-Learn` syntax, as well as the specific methods `metric()`, `transformer()` and `metadata()` that allow us to access the learned metric matrix, the learned linear map or several metadata generated during the learning process, respectively.

It is important to emphasize that these algorithms include different hyperparameters that can be modified to improve the performance or to change the conditions of the learned distances. To this end the package includes `tune` functions, which allow the parameters of the DML algorithms to be easily estimated with cross validation, using the success rate of a k-neighbors classifier or some of the metadata of the algorithms as validation metrics. A detailed description of all hyperparameters for each algorithm can be found in the pyDML's full documentation[4].

[3] `http://scikit-learn.org/stable/modules/generated/sklearn.base.TransformerMixin.html#sklearn.base.TransformerMixin`

[4] `https://pydml.readthedocs.io/`

The pyDML library also incorporates graphical tools for the representation and evaluation of the learned distances, which use the `Matplotlib` library internally. These tools allow labeled data to be represented, along with the regions determined by any `Scikit-Learn` classifier, including distance-based classifiers, for which several functionalities are provided to easily represent the effect of different distances.

3 Installation and Usage

The pyDML library can be installed through `PyPI` (*Python package index*), using the command `pip install pyDML`. It is also possible to download or clone the repository directly from GitHub. In such a case, the installation of the software package can be done by running the setup script available in the root directory, using the command `python setup.py install`. Once installed, we can access all DML algorithms, and the additional functionalities, by importing the desired class within the `dml` module.

As already mentioned, the way DML algorithms are used is similar to the `Scikit-Learn` transformers. Figure 1 shows a basic example. More detailed examples of all the posibilities offered by pyDML can be found in the documentation[5].

```
>>> import numpy as np                    # NumPy library
>>> from sklearn.datasets import load_iris # Iris dataset
>>> from dml import NCA           # Loading DML algorithm

>>> # Loading dataset
>>> iris = load_iris()
>>> X = iris['data']
>>> y = iris['target']

>>> nca = NCA()  # DML construction
>>> nca.fit(X,y) # Fitting algorithm

>>> # We can look at the algorithm metadata
>>> # after fitting it
>>> meta = nca.metadata()
>>> meta
{'final_expectance': 0.95771240234375,
 'initial_expectance': 0.8380491129557291,
 'num_iters': 3}
>>> # We can see the metric the algorithm has learned.
>>> M = nca.metric()
```

```
>>> M
array([[ 1.1909,  0.5129, -2.1581, -2.0146],
       [ 0.5129,  1.5812, -2.1457, -2.1071],
       [-2.1581, -2.1457,  6.4688,  5.8628],
       [-2.0146, -2.1071,  5.8628,  6.8327]])
>>> # Equivalently, we can see the learned linear map.
>>> L = nca.transformer()
>>> L
array([[ 0.7796, -0.0191, -0.3586, -0.2399],
       [-0.0444,  1.0074, -0.2993, -0.2581],
       [-0.6074, -0.5728,  2.1609,  1.3521],
       [-0.4606, -0.4875,  1.2573,  2.2091]])
>>> # Finally, we can obtain the transformed data,
>>> # or transform new data.
>>> Lx = nca.transform() # Transforming training set.
>>> Lx[:5,:]
array([[ 3.3590,  2.8288, -1.8073, -1.8538],
       [ 3.2126,  2.3339, -1.3993, -1.5179],
       [ 3.0887,  2.5743, -1.6085, -1.6490],
       [ 2.9410,  2.4181, -1.0583, -1.3027],
       [ 3.2791,  2.9340, -1.8038, -1.8565]])
```

Figure 1: Use of distance metric learning algorithms in pyDML.

4 Quality Standards

The project code follows the PEP8 style standards for Python code. Continuous integration is performed, using the `Travis CI` service, to ensure back-compatibility and integrate code

[5] `https://pydml.readthedocs.io/en/latest/examples.html`

in a simple way. The project adheres to *Semantic Versioning* and uses the *Keep a Changelog* standards to make it easier to users to see the changes between each version. A thorough documentation is provided using `sphinx` and `numpydoc`, and is hosted in the *Read the Docs* platform. Finally, a stats website is also provided[6], where the performance of the implemented algorithms is evaluated under different conditions [2]. Those algorithms available in other DML libraries are also compared to each other in this website.

5 Conclusions and Future Work

In this paper, we presented a new Python library that integrates a wide range of distance metric learning algorithms, with additional functionalities such as visualization or parameter estimation. The `pyDML` library is fully compatible with Scikit-Learn and is distributed under GPLv3 license.

As future work we plan to extend the library by adding more recent algorithms, and also algorithms oriented to problems beyond standard classification, like ordinal classification [24], imbalanced classification [25] or non-standard problems [26]. We will also explore new ways of learning distances beyond the Mahalanobis approach, such as *deep metric learning* [27].

Acknowledgements

Our work has been supported by the research project TIN2017-89517-P and by a research scholarship (FPU18/05989), given to the author Juan Luis Suárez by the Spanish Ministry of Science, Innovation and Universities.

References

[1] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[2] Juan Luis Suárez, Salvador García, and Francisco Herrera. A tutorial on distance metric learning: Mathematical foundations, algorithms and experiments. *arXiv preprint arXiv:1812.05944*, 2019.

[3] A. Bellet, A. Habrard, and M. Sebban. *Metric Learning*. Morgan & Claypool, 2015.

[4] B. Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2013.

[5] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[6] `https://jlsuarezdiaz.github.io/software/pyDML/stats/index.html#`

[6] Yiming Ying and Peng Li. Distance metric learning with eigenvalue optimization. *Journal of Machine Learning Research*, 13(Jan):1–26, 2012.

[7] Yuan Tang, Tao Gao, and Nan Xiao. dml: Distance metric learning in r. *Journal of Open Source Software*, 3(30):1036, 2018.

[8] William de Vazelhes, Cj Carey, Yuan Tang, Nathalie Vauquier, and Aurélien Bellet. metric-learn: Metric learning algorithms in python. *arXiv preprint arXiv:1908.04710*, 2019.

[9] I.T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, 2002.

[10] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[11] Masashi Sugiyama. Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. *Journal of Machine Learning Research*, 8(May):1027–1061, 2007.

[12] Fei Wang and Changshui Zhang. Feature extraction by maximizing the average neighborhood margin. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[13] Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Ruslan R Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, pages 513–520, 2005.

[14] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2624–2637, 2013.

[15] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine learning*, pages 209–216. ACM, 2007.

[16] Bac Nguyen, Carlos Morell, and Bernard De Baets. Supervised distance metric learning through maximization of the jeffrey divergence. *Pattern Recognition*, 64:215–225, 2017.

[17] Amir Globerson and Sam T Roweis. Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems*, pages 451–458, 2006.

[18] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems*, pages 521–528, 2003.

[19] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Is that you? metric learning approaches for face identification. In *2009 IEEE 12th International Conference on Computer Vision*, pages 498–505. IEEE, 2009.

[20] Pourya Zadeh, Reshad Hosseini, and Suvrit Sra. Geometric mean metric learning. In *International Conference on Machine Learning*, pages 2464–2471, 2016.

[21] Sebastian Mika, Gunnar Ratsch, Jason Weston, Bernhard Scholkopf, and Klaus-Robert Mullers. Fisher discriminant analysis with kernels. In *Neural networks for signal processing IX, 1999. Proceedings of the 1999 IEEE signal processing society workshop.*, pages 41–48. IEEE, 1999.

[22] Lorenzo Torresani and Kuang-Chih Lee. Large margin component analysis. In *Advances in Neural Information Processing Systems*, pages 1385–1392, 2007.

[23] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

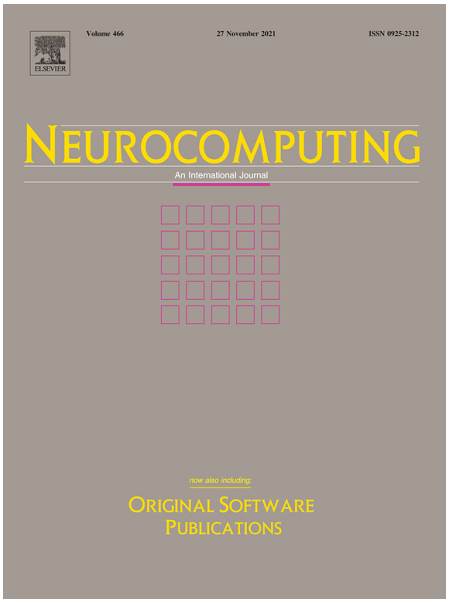
[24] Bac Nguyen, Carlos Morell, and Bernard De Baets. Distance metric learning for ordinal classification based on triplet constraints. *Knowledge-Based Systems*, 142:17–28, 2018.

[25] Nan Wang, Xibin Zhao, Yu Jiang, and Yue Gao. Iterative metric learning for imbalance data classification. In *International Joint Conferences on Artificial Intelligence*, pages 2805–2811, 2018.

[26] David Charte, Francisco Charte, Salvador García, and Francisco Herrera. A snapshot on nonstandard supervised learning problems: taxonomy, relationships, problem transformations and algorithm adaptations. *Progress in Artificial Intelligence*, 8(1):1–14, 2019.

[27] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Deep metric learning for person re-identification. In *2014 IEEE 22nd International Conference on Pattern Recognition*, pages 34–39. IEEE, 2014.

2 A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges



- **Journal:** Neurocomputing (NEUCOM)
- **JCR Impact Factor:** 5.779
- **Rank:** 39/145
- **Quartile:** Q2
- **Category:** Computer Science, Artificial Intelligence
- **Status:** Published

Ref.: Suárez, J. L., García, S. & Herrera, F. (2021). A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges. Neurocomputing, 425, 300-322. DOI: https://doi.org/10.1016/j.neucom.2020.08.017

A TUTORIAL ON DISTANCE METRIC LEARNING: MATHEMATICAL FOUNDATIONS, ALGORITHMS, EXPERIMENTAL ANALYSIS, PROSPECTS AND CHALLENGES

Juan Luis Suárez [*,a,b] **Salvador García** [a,b] **Francisco Herrera** [a,b]

[a] *Department of Computer Science and Artificial Intelligence (DECSAI), University of Granada, Granada, Spain*
[b] *Andalusian Institute of Data Science and Computational Intelligence (DaSCI)*

ABSTRACT

Distance metric learning is a branch of machine learning that aims to learn distances from the data, which enhances the performance of similarity-based algorithms. This tutorial provides a theoretical background and foundations on this topic and a comprehensive experimental analysis of the most-known algorithms. We start by describing the distance metric learning problem and its main mathematical foundations, divided into three main blocks: convex analysis, matrix analysis and information theory. Then, we will describe a representative set of the most popular distance metric learning methods used in classification. All the algorithms studied in this paper will be evaluated with exhaustive testing in order to analyze their capabilities in standard classification problems, particularly considering dimensionality reduction and kernelization. The results, verified by Bayesian statistical tests, highlight a set of outstanding algorithms. Finally, we will discuss several potential future prospects and challenges in this field. This tutorial will serve as a starting point in the domain of distance metric learning from both a theoretical and practical perspective.

Keywords Distance Metric Learning · Classification · Mahalanobis Distance · Dimensionality · Similarity.

* Corresponding Author (jlsuarezdiaz@decsai.ugr.es)
Email addresses: salvagl@decsai.ugr.es (Salvador García), herrera@decsai.ugr.es (Francisco Herrera)

1 Introduction

The use of distances in machine learning has been present since its inception. Distances provide a similarity measure between the data, so that close data will be considered similar, while remote data will be considered dissimilar. One of the most popular examples of similarity-based learning is the well-known nearest neighbors rule for classification, where a new sample is labeled with the majority class within its nearest neighbors in the training set. This classifier was presented in 1967 by [1], even though this idea had already been mentioned in earlier publications [2, 3].

Algorithms in the style of the nearest neighbors classifier are among the main motivators of distance metric learning. These kinds of algorithms have usually used a standard distance, like the Euclidean distance, to measure the data similarity. However, a standard distance may ignore some important properties available in our dataset, and therefore the learning results might be non optimal. The search for a distance that brings similar data as close as possible, while moving non similar data away, can significantly increase the quality of these algorithms.

During the first decade of the 21st century some of the most well-known distance metric learning algorithms were developed, and perhaps the algorithm from [4] is responsible for drawing attention to this concept for the first time. Since then, distance metric learning has established itself as a promising domain in machine learning, with applications in many fields, such as medicine [5, 6], security [7, 8], social media mining [9, 10], speech recognition [11, 12], information retrieval [13, 14], recommender systems [15, 16] and many areas of computer vision, such as person re-identification [17, 18], kinship verification [19, 20] or image classification [21, 22].

Although distance metric learning has proven to be an alternative to consider with small and medium datasets [23], one of its main limitations is the treatment of large data sets, both at the level of number of samples and at the level of number of attributes [24]. In recent years, several alternatives have been explored in order to develop algorithms for these areas [25, 26].

Several surveys on distance metric learning have been proposed. Among the well-known surveys we can find the work of [27], [28], [29] and [30]. However, we must point out several 'loose ends' present in these studies. On the one hand, they do not provide an in-depth study of the theory behind distance metric learning. Such a study would help to understand the motivation behind the mechanics that give rise to the various problems and tools in this field. On the other hand, previous studies do not carry out enough experimental analyses that evaluate the performance of the algorithms on sufficiently diverse datasets and circumstances.

In this paper we undergo a theoretical study of supervised distance metric learning, in which we show the mathematical foundations of distance metric learning and its algorithms. We analyze several distance metric learning algorithms for classification, from the problems and the objective functions they try to optimize, to the methods that lead to the solutions to these

problems. Finally, we carry out several experiments involving all the algorithms analyzed in this study. In our paper, we want to set ourselves apart from previous publications by focusing on a deeper analysis of the main concepts of distance metric learning, trying to get to its basic ideas, as well as providing an experimental framework with the most popular metric learning algorithms. We will also discuss some opportunities for future work in this topic.

Regarding the theoretical background of distance metric learning, we have studied three mathematical fields that are closely related to this topic. The first one is convex analysis [31, 32]. Convex analysis is present in many distance metric learning algorithms, as these algorithms they try to optimize convex functions over convex sets. Some interesting properties of convex sets, as well as how to deal with constrained convex problems, will be shown in this study. We will also see how the use of matrices is a fundamental part when modeling our problem. Matrix analysis [33] will therefore be the second field studied. The third is information theory [34], which is also used in some of the algorithms we will present.

As explained before, our work focuses on supervised distance metric learning techniques. A large number of algorithms have been proposed over the years. These algorithms were developed with different purposes and based on different ideas, so that they could be classified into different groups. Thus, we can find algorithms whose main goal is dimensionality reduction [35, 36, 37], algorithms specifically oriented to improve distance based classifiers, such as the nearest neighbors classifier [38, 39], or the nearest centroid classification [40], and a few techniques that are based on information theory [41, 42, 43]. Some of these algorithms also allow kernel versions [44, 45, 36, 42], that allow for the extension of distance metric learning to highly dimensional spaces.

As can be seen in the experiments, we compare all the studied algorithms using up to 34 different datasets. In order to do so, we define different settings to explore their performance and capabilities when considering maximum dimension, centroid-based methods, different kernels and dimensionality reduction. Bayesian statistical tests are used to assess the significant differences among algorithms [46].

In summary, the aims of this tutorial are:

- To know and understand the discipline of distance metric learning and its foundations.
- To gather and study the foundations of the main supervised distance metric learning algorithms.
- To provide experiments to evaluate the performance of the studied algorithms under several case studies and to analyze the results obtained. The code of the algorithms is available in the Python library `pydml` [47].

So as to avoid overloading the paper with a large theoretical content a publicly downloadable theoretical supplement is provided as appendix of the current public draft in ArXiv [48]. In

this document Appendix B presents in a rigorous and detailed manner the mathematical foundations of distance metric learning, structured in the three blocks discussed previously, and Appendix C provides a detailed explanation of the algorithms.

Our paper is organized as follows. Section 2 introduces the distance metric problem and its mathematical foundations, explains the family of distances we will work with and shows several examples and applications. Section 3 discusses all the distance metric learning algorithms chosen for this tutorial. Section 4 describes the experiments done to evaluate the performance of the algorithms and shows the obtained results. Finally, Sections 5 and 6 conclude the paper by indicating possible future lines of research in this area and summarizing the work done, respectively. We also provide a glossary of terms at the end of the document (A) with the acronyms used in the paper.

2 Distance Metric Learning and Mathematical Foundations

In this section we will introduce the distance metric learning problem. To begin with, we will provide reasons to support the distance metric learning problem in Section 2.1. Then, we will go over the concept of distance, with special emphasis on the Mahalanobis distances, which will allow us to model our problem (Section 2.2). Once these concepts are defined, in Section 2.3 we will describe the distance metric learning problem, explaining what it consists of, how it is handled in the supervised case and how it is modeled so that it can be treated computationally. To understand the basics of distance metric learning we provide a summary of the mathematical foundations underlying this field in Section 2.4. These foundations support the theoretical description of this discipline as well as the numerous distance metric learning algorithms that will be discussed in Section 3 and [48, Appendix C]. The mathematical background is then developed extensively in [48, Appendix B]. Finally, we will finish with Section 2.5 by detailing some of the uses of distance metric learning in machine learning.

2.1 Distance Metric Learning: Why and What For?

Similarity-based learning algorithms are among the earliest used in the field of machine learning. They are inspired by one of the most important components in many human cognitive processes: the ability to detect similarities between different objects. This ability has been adapted to machine learning by designing algorithms that learn from a dataset according to the similarities present between the data. These algorithms are present in most areas of machine learning, such as classification and regression, with the k-Nearest Neighbors (k-NN) rule [1]; in clustering, with the k-means algorithm [49]; in recommender systems, with collaborative approaches based also on nearest neighbors [50]; in semi-supervised learning, to construct the graph representations [51]; in some kernel methods such as the radial basis functions [52], and many more.

To measure the similarity between data, it is necessary to introduce a distance, which allows us to establish a measure whereby it is possible to determine when a pair of samples is more similar than another pair of samples. However, there is an infinite number of distances we

can work with, and not all of them will adapt properly to our data. Therefore, the choice of an adequate distance is a crucial element in this type of algorithm.

Distance metric learning arises to meet this need, providing algorithms that are capable of searching for distances that are able to capture features or relationships hidden in our data, which possibly a standard distance, like the Euclidean distance, would not have been able to discover. From another perspective, distance metric learning can also be seen as the missing training step in many similarity-based algorithms, such as the *lazy* nearest-neighbor approaches. The combination of both the distance learning algorithms and the distance-based learners allows us to build more complete learning algorithms with greater capabilities to extract information of interest from our data.

Choosing an appropriate distance learned from the data has proven to be able to greatly improve the results of distance-based algorithms in many of the areas mentioned above. In addition to its potential when adhering to these learners, a good distance allows data to be transformed to facilitate their analysis, with mechanisms such as dimensionality reduction or axes selection, as we will discuss later.

2.2 Mahalanobis Distances

We will start by reviewing the concept of distance and some of its properties.

Definition 1. *Let X be a non-empty set. A* distance *or* metric *over X is a map $d : X \times X \to \mathbb{R}$ that satisfies the following properties:*

1. *Coincidence: $d(x, y) = 0 \iff x = y$, for every $x, y \in X$.*
2. *Symmetry: $d(x, y) = d(y, x)$, for every $x, y \in X$.*
3. *Triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$, for every $x, y, z \in X$.*

The ordered pair (X, d) is called a metric space.

The coincidence property stated above will not be of importance to us. That is why we will also consider mappings known as *pseudodistances*, which only require that $d(x, x) = 0$, instead of the coincidence property. In fact, pseudodistances are strongly related with dimensionality reduction, which is an important application of distance metric learning. From now on, when we talk about distances, we will be considering proper distances as well as pseudodistances.

Remark. *As an immediate consequence of the definition, we have the following additional properties of distances:*

4. *Non negativity: $d(x, y) \geq 0$ for every $x, y \in X$.*
5. *Reverse triangle inequality: $|d(x, y) - d(y, z)| \leq d(x, z)$ for every $x, y, z \in X$.*
6. *Generalized triangle inequality: $d(x_1, x_n) \leq \sum_{i=1}^{n-1} d(x_i, x_{i+1})$ for $x_1, \dots, x_n \in X$.*

When we work in the d-dimensional Euclidean space, a family of distances become very useful in the computing field. These distances are parameterized by positive semidefinite matrices and are known as *Mahalanobis distances*. In what follows, we will refer to $\mathcal{M}_{d'\times d}(\mathbb{R})$ (resp. $\mathcal{M}_d(\mathbb{R})$) as the set of matrices of dimension $d'\times d$ (resp. square matrices of dimension d), and to $S_d(\mathbb{R})_0^+$ as the set of positive semidefinite matrices of dimension d.

Definition 2. *Let $d \in \mathbb{N}$ and $M \in S_d(\mathbb{R})_0^+$. The* Mahalanobis distance *corresponding to the matrix M is the map $d_M : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ given by*

$$d_M(x,y) = \sqrt{(x-y)^T M(x-y)}, \quad x,y \in \mathbb{R}^d.$$

Mahalanobis distances come from the (semi-)dot products in $\mathbb{R}^d$ defined by the positive semidefinite matrix M. When M is full-rank, Mahalanobis distances are proper distances. Otherwise, they are pseudodistances. Note that the Euclidean usual distance is a particular example of a Mahalanobis distance, when M is the identity matrix I. Mahalanobis distances have additional properties specific to distances over normed spaces.

7. Homogeneousness: $d(ax,ay) = |a|d(x,y)$, for $a \in \mathbb{R}$, and $x,y \in \mathbb{R}^d$.
8. Translation invariance: $d(x,y) = d(x+z,y+z)$, for $x,y,z \in \mathbb{R}^d$.

Sometimes the term "Mahalanobis distance" is used to describe the squared distances of the form $d_M^2(x,y) = (x-y)^T M(x-y)$. In the area of computing, it is much more efficient to work with d_M^2 rather than with d_M, as this avoids the calculation of square roots. Although d_M^2 is not really a distance, it keeps the most useful properties of d_M from the distance metric learning perspective, as we will see, such as the greater or lesser closeness between different pairs of points. That is why the use of the term "Mahalanobis distance" for both d_M and d_M^2 is quite widespread.

To end this section, we return to the issue of dimensionality reduction that we mentioned when introducing the concept of pseudodistance. When we work with a pseudodistance σ over a set X, it is possible to define an equivalence relationship given by $x \sim y$ if and only if $\sigma(x,y) = 0$, for each $x,y \in X$. Using this relationship we can consider the quotient space $X/_\sim$, and the map $\hat{\sigma} : X/_\sim \times X/_\sim \to \mathbb{R}$ given by $\hat{\sigma}([x],[y]) = \sigma(x,y)$, for each $[x],[y] \in X/_\sim$. This map is well defined and is a distance over the quotient space. When σ is a Mahalanobis distance over $\mathbb{R}^d$, with rank $d' < d$ (we define the rank of a Mahalanobis distance as the rank of the associated positive semidefinite matrix matrix), then the previous quotient space becomes a vector space isomorphic to $\mathbb{R}^{d'}$, and the distance $\hat{\sigma}$ is a full-rank Mahalanobis distance over $\mathbb{R}^{d'}$. That is why, when we have a Mahalanobis pseudodistance on $\mathbb{R}^d$, we can view this as a proper Mahalanobis distance over a lower dimensional space, hence we have obtained a dimensionality reduction.

2.3 Description of Distance Metric Learning

Distance Metric Learning (DML) is a machine learning discipline with the purpose of learning distances from a dataset. In its most general version, a dataset $\mathcal{X} = \{x_1, \dots, x_N\}$ is avail-

able, on which certain similarity measures between different pairs or triplets of data are collected. These similarities are determined by the sets

$$\begin{aligned} S &= \{(x_i, x_j) \in \mathcal{X} \times \mathcal{X} : \ x_i \text{ and } x_j \text{ are similar.}\}, \\ D &= \{(x_i, x_j) \in \mathcal{X} \times \mathcal{X} : \ x_i \text{ and } x_j \text{ are not similar.}\}, \\ R &= \{(x_i, x_j, x_l) \in \mathcal{X} \times \mathcal{X} \times \mathcal{X} : \ x_i \text{ is more similar to } x_j \text{ than to } x_l.\}. \end{aligned}$$

With these data and similarity constraints, the problem to be solved consists in finding, after establishing a family of distances $\mathcal{D}$, those distances that best adapt to the criteria specified by the similarity constraints. To do this, a certain loss function ℓ is set, and the sought-after distances will be those that solve the optimization problem

$$\min_{d \in \mathcal{D}} \ell(d, S, D, R).$$

When we focus on supervised learning, in addition to dataset $\mathcal{X}$ we have a list of labels $y_1, \dots, y_N$ corresponding to each sample in $\mathcal{X}$. The general formulation of the DML problem is easily adapted to this new situation, just by considering the sets S and D as sets of pairs of same-class samples and different-class samples, respectively. Two main approaches are followed to establish these sets. The *global DML* approach considers the sets S and D to be

$$\begin{aligned} S &= \{(x_i, x_j) \in \mathcal{X} \times \mathcal{X} : \ y_i = y_j\}, \\ D &= \{(x_i, x_j) \in \mathcal{X} \times \mathcal{X} : \ y_i \neq y_j\}. \end{aligned}$$

On the other hand, the *local DML* approach replaces the previous definition of S with

$$S = \{(x_i, x_j) \in \mathcal{X} \times \mathcal{X} : \ y_i = y_j \text{ and } x_j \in \mathcal{U}(x_i)\},$$

where $\mathcal{U}(x_i)$ denotes a *neighborhood* of x_i, that is, a set of points that should be close to x_i, which has to be established before the learning process by using some sort of prior information, or a standard similarity measure. The set D remains the same in the local approach, since different-class samples are not meant to be similar in a supervised learning setting. In addition, the set R may be also available in both approaches by defining triplets (x_i, x_j, x_l) where in general $y_i = y_j \neq y_l$, and they verify certain conditions imposed on the distance between x_i and x_j, as opposed to the distance between x_i and x_l. This is the case, for example, for impostors in the LMNN algorithm (see Section 3.2.1 and [38]). In any case, labels have all the necessary information in the field of supervised DML. From now on we will focus on this kind of problem.

Furthermore, focusing on the nature of the dataset, practically all of the DML theory is developed for numerical data. Although it is possible to define relevant distances for non-numerical attributes [53, 54] and although some learning processes can be performed with them [55, 56], the richness of the distances available to numerical features, their ability to be parameterized computationally, and the fact that nominal data can be converted to numerical variables or ordinal variables, with appropriate encoding [57], cause the relevant

distances in this discipline to be those defined for numerical data. For this reason, from now on, we will focus on supervised learning problems with numerical datasets.

We will suppose then that $\mathcal{X} \subset \mathbb{R}^d$. As we saw in the previous section, for finite-dimensional vector spaces we have the family of Mahalanobis distances, $\mathcal{D} = \{d_M : M \in S_d(\mathbb{R})_0^+\}$. With this family, we have at our disposal all the distances associated with dot products in $\mathbb{R}^d$ (and in lower dimensions). In addition, this family is determined by the set of positive semidefinite matrices, and therefore, we can use these matrices, which we will call *metric matrices*, to parameterize distances. In this way, the general problem adapted to supervised learning with Mahalanobis distances can be rewritten as

$$\min_{M \in S_d(\mathbb{R})_0^+} \ell(d_M, (x_1, y_1), \dots, (x_N, y_N)).$$

However, this is not the only way to parameterize this type of problem. We know, from the *matrix decomposition theorem* discussed in Section 2.4 and [48, Theorem 5], that if $M \in S_d(\mathbb{R})_0^+$, then there is a matrix $L \in \mathcal{M}_d(\mathbb{R})$ so that $M = L^T L$, and this matrix is unique except for an isometry. So, then we get

$$d_M^2(x, y) = (x - y)^T M (x - y) = (x - y)^T L^T L (x - y) = (L(x - y))^T (L(x - y)) = \|L(x - y)\|_2^2.$$

Therefore, we can also parameterize Mahalanobis distances through any matrix, although in this case the interpretation is different. When we learn distances through positive semidefinite matrices we are learning a new metric over $\mathbb{R}^d$. When we learn distances with the previous L matrices, we are learning a linear map (given by $x \mapsto Lx$) that transforms the data in the space, and the corresponding distance is the usual Euclidean distance after projecting the data onto the new space using the linear map. Both approaches are equivalent thanks to the matrix decomposition theorem [48, Theorem 5].

In relation to dimensionality, it is important to note that, when the learned metric M is not full-rank, we are actually learning a distance over a space of lower dimension (as we mentioned in the previous section), which allows us to reduce the dimensionality of our dataset. The same occurs when we learn linear maps that are not full-rank. We can extend this case and opt to learn directly linear maps defined by $L \in \mathcal{M}_{d' \times d}(\mathbb{R})$, with $d' < d$. In this way, we ensure that data are directly projected into a space of dimension no greater than d'.

Both learning the metric matrix M and learning the linear transformation L, are useful approaches to model DML problems, each one with its advantages and disadvantages. For example, parameterizations via M usually lead to convex optimization problems. In contrast, convexity in problems parameterized by L is not so easy to achieve. On the other hand, parameterizations using L make it possible to learn projections directly onto lower dimensional spaces, while dimensional constraints for problems parameterized by M are not so easy to achieve. Let us examine these differences with simple examples.

Example. *Many of the functions we will want to optimize will depend on the squared distance defined by the metric M or by the linear transformation L, that is, either they will have terms of the form $\|v\|_M^2 = v^T M v$, or of the form $\|v\|_L^2 = \|Lv\|_2^2$. Both the maps $M \mapsto \|v\|_M^2$ and*

$L \mapsto \|v\|_L^2$ are convex (the first is actually affine). However, if we want to substract terms in this way, we lose convexity in L, because the mapping $L \mapsto -\|v\|_L^2$ is no longer convex. In contrast, the mapping $M \mapsto -\|v\|_M^2$ is still affine and, therefore, convex.

Example. *Rank constraints are not convex, and therefore we may not dispose of a projection onto the set corresponding to those constraints, unless we learn the mapping (parameterized by L) directly to the space with the desired dimension, as explained before. For example, if we consider the set $C = \{M \in S_2(\mathbb{R})_0^+ : r(A) \leq 1\}$, we get $A = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix} \in C$ and $B = \begin{pmatrix} 0 & 0 \\ 0 & 2 \end{pmatrix} \in C$. However, $(1-\lambda)A + \lambda B = I \notin C$, for $\lambda = 1/2$.*

2.4 Mathematical Foundations of Distance Metric Learning

There are three main mathematical areas that support DML: convex analysis, matrix analysis and information theory. The first provides the necessary tools so that many of the algorithms can address their optimization problems. Thanks to the second we can parameterize DML, and in this way we can compute the different problems. It also provides us with some interesting results when it comes to solving certain problems related to dimensionality reduction. Finally, the third field provides us with concepts and tools that are very useful for designing algorithms that use probability distributions associated with the data.

2.4.1 Convex Analysis

Let us begin with convex analysis. One of the properties of convex sets that makes convex analysis of great interest in DML is known as the *convex projection theorem* [48, Theorem 2], which ensures that for any non-empty convex closed set K in $\mathbb{R}^d$ and for every point $x \in \mathbb{R}^d$ there is a single point $x_0 \in K$ for which the distance from x to K is the same to the distance from x to x_0. That is, the distance from x to K is materialized in the point x_0, which is called the *projection of x to K*.

The existence of a projection mapping onto any closed and convex set in $\mathbb{R}^d$ is fundamental when optimizing convex functions with convex constraints, which are frequent, in particular, in many DML algorithms. Let us first discuss optimization mechanisms when working with unconstrained differentiable functions, which, although they do not strictly take part in convex analysis, are also present in some DML algorithms and are the basis for convex optimization mechanisms. In these cases, the most popular techniques are the well-known *gradient descent methods*, which are iterative methods. The basic idea of gradient descent methods is to move in the direction of the gradient of the objective function in order to optimize it. We show in [48, Appendix B.1.2] that, indeed, small displacements in the gradient direction guarantee the improvement of the objective function, proving the effectiveness of these methods.

Returning to the constrained case, we can see that gradient descent methods are no longer valid, since the displacement in the gradient direction can no longer fulfill the constraints. However, we will show that, if after the gradient step we project the obtained point onto the convex set determined by the constraints, the combination of both movements contributes

to improving the value of the objective function as long as the initial gradient step is small enough. This extension of gradient descent to the constrained convex case is known as the *projected gradient method*. There are also other approaches, such as the penalty methods, which allow these problems to be handled by transforming the constrained objective function into a new unconstrained objective function in which the violations of the previous constraints are converted into penalties that worsen the value of the new objective function [58]. They will not usually, however, be preferred in the matrix problems we will be dealing with, as they may be computationally expensive and difficult to adapt [59].

Finally, we must highlight other tools of interest for the optimization problems to be studied. Firstly, when working with convex problems with multiple constraints, the projections on each individual constraint are often known, but the projection onto the set determined by all the constraints is not. With the method known as the *iterated projections method* [48, Appendix B.1.2] we can approach this projection by subsequently projecting onto each of the individual constraints until convergence (which is guaranteed) is obtained. Lastly, convex functions that are not differentiable everywhere can still be optimized following the approaches discussed here, as they admit sub-gradients at every point. Sub-gradient descent methods [48, Appendix B.1.2] can work in the same way as gradient descent methods and can therefore be applied with convex functions that may not be differentiable at some points.

2.4.2 Matrix Analysis

As we have already seen, matrices are a key element in DML. There are several results that are essential for the development of the DML theory and its algorithms. The first of these is the *matrix decomposition theorem* [48, Theorem 5], which was already mentioned in Section 2.3. This theorem states that for any positive matrix $M \in S_d(\mathbb{R})_0^+$ there is a matrix $L \in \mathcal{M}_d(\mathbb{R})$ so that $M = L^T L$ and L is unique except for an isometry. This result allows us to approach DML from the two perspectives (learning the metric M or learning the linear map L) already discussed in Section 2.3.

An important aspect when designing DML algorithms is the geometric manipulation of the matrices (for learning both M and L). Observe that to be able to talk about the convex analysis concepts discussed previously over the set of matrices, we first need to establish an inner product over them. The *Frobenius inner product* allows us to identify matrices as vectors where we add the matrix rows one after the other, and then compute the usual vectorial inner product with these vectors. With the Frobenius product we convert the matrices set in a Hilbert space, and therefore can apply the convex analysis theory studied in the previous section.

Staying on this subject, we have to highlight a case study of particular interest. We will see many situations where we want to optimize a convex function defined on a matrix space, with the restriction that the variable is positive semidefinite. These optimization problems are convex and are usually called *semidefinite programming* problems. We can optimize these objective functions using the projected gradient descent method. The *semidefinite projection theorem* [48, Theorem 4] states that we can compute the projection of a matrix

onto the positive semidefinite cone by performing an eigenvalue decomposition, nullifying the negative eigenvalues and recomposing the matrix with the new eigenvalues. Therefore, we know how to project onto the constraint set and consequently we can apply the projected gradient method.

Finally, we will see that certain algorithms, especially those associated with dimensionality reduction, use optimization problems with similar structures. These problems involve one or more symmetric matrices and the objective function is obtained as a trace after performing certain operations with these matrices. This is, for instance, the case of the objective function of the well-known *principal components analysis*, which can be written as

$$\begin{aligned} \max_{L \in \mathcal{M}_{d' \times d}(\mathbb{R})} \quad & \operatorname{tr}\left(LAL^T\right) \\ \text{s.t.:} \quad & LL^T = I, \end{aligned}$$

where A is a symmetric matrix of dimension d. These problems have the property that they can be optimized without using gradient methods, since an optimum can be built by taking the eigenvectors associated with the largest eigenvalues of the matrices involved in the problem (in the case described here, the eigenvectors of A). In [48, Appendix B.2.3] we present these problems in more detail and show how their solution is obtained.

2.4.3 Information Theory

Information theory is a branch of mathematics and computer theory with the purpose of establishing a rigorous measure to quantify the information and disorder found in a communication message. It has been applied in most science and engineering fields. In DML, information theory is used in several algorithms to measure the closeness between probability distributions. Then, these algorithms try to find a distance metric for which these probability distributions are as close as possible or as far as possible, depending on what distributions are defined. The measures used in this area, unlike distances, only require the properties of non-negativity and coincidence, and are called *divergences*. We will use two different divergences throughout this study:

- The *relative entropy* or the *Kullback-Leibler divergence*, defined for probability distributions p and q, and X the random variable corresponding to p as

$$\mathrm{KL}(p\|q) = \mathbb{E}_p\left[\log \frac{p(X)}{q(X)}\right].$$

- The *Jeffrey divergence* or the *symmetric relative entropy*, defined for p, q and X in the same conditions as above, as

$$\mathrm{JF}(p\|q) = \mathrm{KL}(p\|q) + \mathrm{KL}(q\|p).$$

The key fact that makes divergences very useful in DML is that, when the distributions involved are multivariate gaussian, these divergences can be expressed in terms of matrix divergences, which give rise to problems that can be dealt with quite effectively using the tools

described in this section. In [48, Appendix B.3] we present the matrix expressions obtained for the Kullback-Leibler and the Jeffrey divergences for the most remarkable cases.

2.5 Use Cases in Machine Learning

This section describes some of the most prominent uses of DML in machine learning, illustrated with several examples.

- **Improve the performance of distance-based classifiers.** This is one of the main purposes of DML. Using such learning, a distance that fits well with the dataset and the classifier can be found, improving the performance of the classifier [38, 39]. An example is shown in Figure 1.

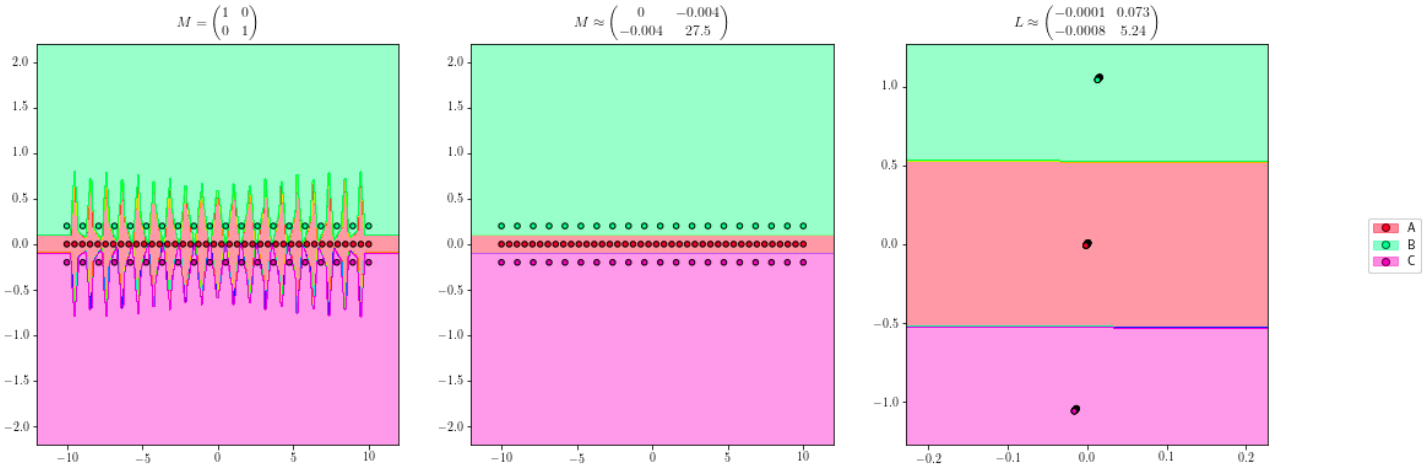


Figure 1: Suppose we have a dataset in the plane, where data can belong to three different classes, whose regions are defined by parallel lines. Suppose that we want to classify new samples using the one nearest neighbor classifier. If we use Euclidean distance, we would obtain the classification regions shown in the image on the left, because there is a greater separation between each sample in class B and class C than there is between the regions. However, if we learn an adequate distance and try to classify with the nearest neighbor classifier again, we obtain much more effective classification regions, as shown in the center image. Finally, as we have seen, learning a metric is equivalent to learning a linear map and to use Euclidean distance in the transformed space. This is shown in the right figure. We can also observe that data are being projected, except for precision errors, onto a line, thus we are also reducing the dimensionality of the dataset.

- **Dimensionality reduction.** As we have already commented, learning a low-rank metric implies a dimensionality reduction on the dataset we are working with. This dimensionality reduction provides numerous advantages, such as a reduction in the computational cost, both in space and time, of the algorithms that will be used later,

or the removal of the possible noise introduced when picking up the data. In addition, some distance-based classifiers are exposed to a problem called *curse of dimensionality* (see, for example, [60], sec. 19.2.2). By reducing the dimension of the dataset, this problem also becomes less serious. Finally, if deemed necessary, projections onto dimension 1, 2 and 3 would allow us to obtain visual representations of our data, as shown in Figure 2. In general, many real-world problems arise with a high dimensionality, and need a dimensionality reduction to be handled properly.

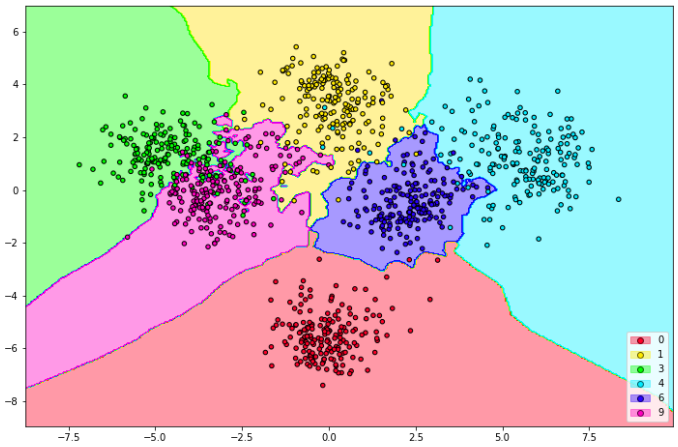


Figure 2: 'Digits' dataset consists of 1797 examples. Each of them consists of a vector with 64 attributes, representing intensity values on an 8x8 image. The examples belong to 10 different classes, each of them representing the numbers from 0 to 9. By learning an appropriate transformation we are able to project most of the classes on the plane, so that we can clearly perceive the differentiated regions associated with each of the classes.

- **Axes selection and data rearrangement.** Closely related to dimensionality reduction, this application is a result of algorithms that learn transformations which allow the coordinate axes to be moved (or selected according to the dimension), so that in the new coordinate system the vectors concentrate certain measures of information on their first components [61]. An example is shown in Figure 3.

- **Improve the performance of clustering algorithms.** Many of the clustering algorithms use a distance to measure the closeness between data, and thus establish the clusters so that data in the same cluster are considered close for that distance. Sometimes, although we do not know the ideal groupings of the data or the number of clusters to establish, we can know that certain pairs of points must be in the same cluster and that other specific pairs must be in different clusters [4]. This happens in numerous problems, for example, when clustering web documents [62]. These documents have a lot of additional information, such as links between documents,

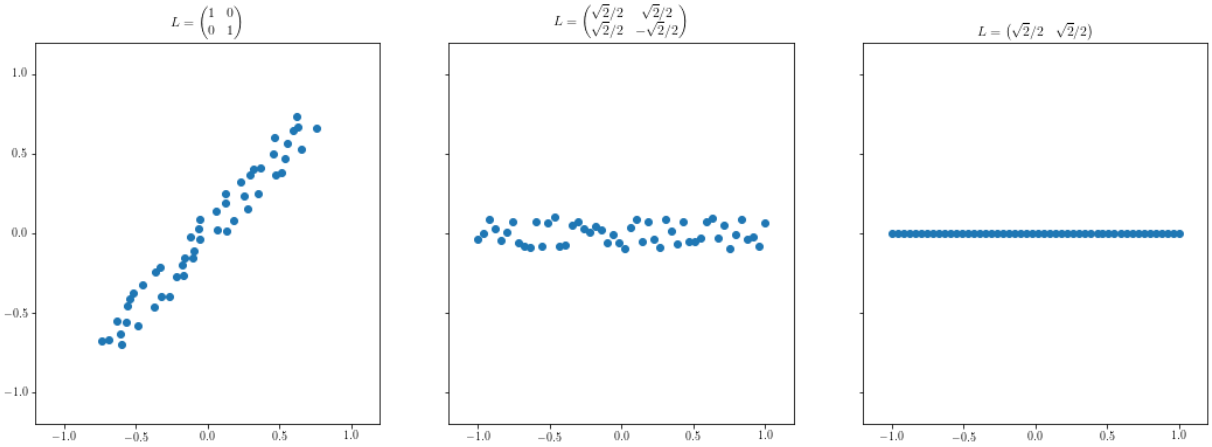


Figure 3: The dataset in the left figure seems to concentrate most of its information on the diagonal line that links the lower left and upper right corners. By learning an appropriate transformation, we can get that direction to fall on the horizontal axis, as shown in the center image. As a result, the first coordinate of the vectors in this new basis concentrates a large part of the variability of the vector. In addition, it seems reasonable to think that the values introduced by the vertical coordinate might be due to noise, and so, we can even just keep the first component, as shown in the right image.

which can be included as similarity constraints. Many clustering algorithms are particularly sensitive to the distance used, although many also depend heavily on the parameters with which they are initialized [63, 64]. It is therefore important to seek a balance or an appropriate aggregation between these two components. In any case, the parameter initialization is beyond the scope of this paper.

- **Semi-supervised learning.** Semi-supervised learning is a learning model in which there is one set of labeled data and another set (generally much larger) of unlabeled data. Both datasets are intended to learn a model that allows new data to be labeled. Semi-supervised learning arises from the fact that in many situations collecting unlabeled data is relatively straightforward, but assigning labels can require a supervisor to assign them manually, which may not be feasible. In contrast, when a lot of unlabeled data is used along with a small amount of labeled data, it is possible to improve learning outcomes considerably, as exemplified in Figure 4. Many of these techniques consist of constructing a graph with weighted edges from the data, where the value of the edges depends on the distances between the data. From this graph we try to infer the labels of the whole dataset, using different propagation algorithms [51]. In the construction of the graph, the choice of a suitable distance is important, thus DML comes into play [65].

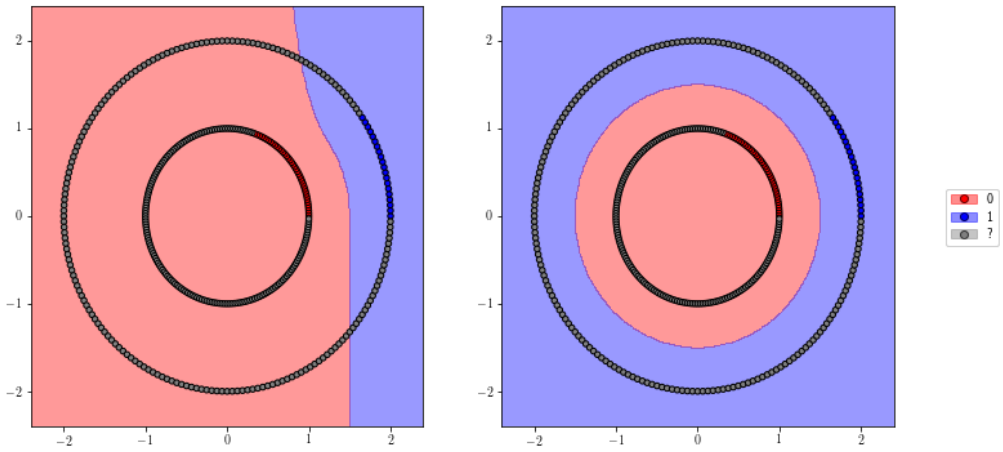


Figure 4: Learning with only supervised information (left) versus learning with all unsupervised information (right).

From the applications we have seen, we can conclude that DML can be viewed as a preprocessing step for many distance-based learning algorithms. The algorithms analyzed in our work focus on the first three applications of the above enumeration. It should be noted that, although the fields above are those where DML has traditionally been used, today, new prospects and challenges for DML are being considered. They will be discussed in Section 5.

3 Algorithms for Distance Metric Learning

This section introduces some of the most popular techniques currently being used in supervised DML. Due to space issues, the section will give a brief description of each of the algorithms, while a detailed description can be found in [48, Appendix C], where the problems the algorithms try to optimize are analyzed, together with their mathematical foundations and the techniques used to solve them.

Table 1 shows the algorithms studied throughout this work, including name, references and a short description. These algorithms will be empirically analyzed in the next section. This study is not intended to be exhaustive and therefore only some of the most popular algorithms have been selected for the theoretical study and the subsequent experimental analysis.

We will now provide a brief introduction to these algorithms. According to the main purpose of each algorithm, we can group them into different categories: dimensionality reduction techniques (Section 3.1), algorithms directed at improving the nearest neighbors classifiers (Section 3.2), algorithms directed at improving the nearest centroid classifiers (Section 3.3),

Name	References	Appendix section [48]	Description
PCA	[66]	C.1.1	A dimensionality reduction technique that obtains directions maximizing variance. Although not supervised, it is important to allow dimensionality reduction in other algorithms that are not able to do so on their own.
LDA	[35]	C.1.2	A dimensionality reduction technique that obtains direction maximizing a ratio involving *between-class* variances and *within-class* variances.
ANMM	[36]	C.1.3	A dimensionality reduction technique that aims at optimizing an average neighborhood margin between same-class neighbors and different-class neighbors, for each of the samples.
LMNN	[38]	C.2.1	An algorithm aimed at improving the accuracy of the k-neighbors classifier. It tries to optimize a two-term error function that penalizes, on the one hand, large distance between each sample and its *target neighbors*, and on the other hand, small distances between different-class samples.
NCA	[39]	C.2.2	An algorithm aimed at improving the accuracy of the k-neighbors classifier, by optimizing the expected *leave-one-out* accuracy for the nearest neighbor classification.
NCMML	[40]	C.3.1	An algorithm aimed at improving the accuracy of the *nearest class mean* classifier, by optimizing a log-likelihood for the labeled data in the training set.
NCMC	[40]	C.3.2	A generalization of the NCMML algorithm aimed at improving nearest centroids classifiers that allow multiple centroids per class.
ITML	[41]	C.4.1	An information theory based technique that aims at minimizing the Kullback-Leibler divergence with respect to an initial gaussian distribution, but while keeping certain similarity constraints between data.
DMLMJ	[42]	C.4.2	An information theory based technique that aims at maximizing the Jeffrey divergence between two distributions, associated to similar and dissimilar points, respectively.
MCML	[43]	C.4.3	An information theory based technique that tries to collapse same-class points in a single point, as far as possible from the other classes collapsing points.
LSI	[4]	C.5.1	A DML algorithm that globally minimizes the distances between same-class points, while fulfilling minimum-distance constraints for different-class points.
DML-eig	[67]	C.5.2	A DML algorithm similar to LSI that offers a different resolution method based on eigenvalue optimization.
LDML	[68]	C.5.3	A probabilistic approach for DML based on the logistic function.
KLMNN	[38]; [44]	C.6.1	The kernel version of LMNN.
KANMM	[36]	C.6.2	The kernel version of ANMM.
KDMLMJ	[42]	C.6.3	The kernel version of DMLMJ.
KDA	[45]	C.6.4	The kernel version of LDA.

Table 1: Description of the DML algorithms analyzed in this study.

or algorithms based on information theory (Section 3.4). These categories are not necessarily exclusive, but we have considered each of the algorithms in the category associated with their dominant purpose. We also introduce, in Section 3.5 several algorithms with less specific goals, and finally, in Section 3.6, the kernel versions for some of the algorithms studied.

We will explain the problems that each of these techniques try to solve. For a more detailed description of each algorithm, the reader can refer to the corresponding section of the appendix supplement [48], as shown in Table 1.

3.1 Dimensionality Reduction Techniques

Dimensionality reduction techniques try to learn a distance by searching a linear transformation from the dataset space to a lower dimensional Euclidean space. We will describe the algorithms PCA [66], LDA [35] and ANMM [36].

3.1.1 Principal Components Analysis (PCA)

PCA [66] is one of the most popular dimensionality reduction techniques in unsupervised DML. Although PCA is an unsupervised learning algorithm, it is necessary to talk about it in our paper, firstly because of its great relevance, and more particularly, because when a supervised DML algorithm does not allow a dimensionality reduction, PCA can be first applied to the data in order to be able to use the algorithm later in the lower dimensional space.

The purpose of PCA is to learn a linear transformation from the original space $\mathbb{R}^d$ to a lower dimensional space $\mathbb{R}^{d'}$ for which the loss when recomposing the data in the original space is minimized. This has been proven to be equivalent to iteratively finding orthogonal directions for which the projected variance of the dataset is maximized. The linear transformation is then the projection onto these directions. The optimization problem can be formulated as

$$\max_{\substack{L\in\mathcal{M}_{d'\times d}(\mathbb{R})\\ LL^T=I}} \operatorname{tr}\left(L\Sigma L^T\right),$$

where Σ is, except for a constant, the covariance matrix of $\mathcal{X}$, and tr is the trace operator. The solution to this problem can be obtained by taking as the rows of L the eigenvectors of Σ associated with its largest eigenvalues.

3.1.2 Linear Discriminant Analysis (LDA)

LDA [35] is a classical DML technique with the purpose of learning a projection matrix that maximizes the separation between classes in the projected space using *within-class* and *between-class* variances. It follows a scheme similar to the one proposed by PCA, but in this case it takes the supervised information provided by the labels into account.

The optimization problem of LDA is formulated as

$$\max_{L \in \mathcal{M}_{d' \times d}(\mathbb{R})} \quad \operatorname{tr}\left((LS_w L^T)^{-1}(LS_b L^T)\right),$$

where S_b and S_w are, respectively, the between-class and within-class *scatter matrices*, which are defined as

$$S_b = \sum_{c \in \mathcal{C}} N_c(\mu_c - \mu)(\mu_c - \mu)^T,$$
$$S_w = \sum_{c \in \mathcal{C}} \sum_{i \in \mathcal{C}_c} (x_i - \mu_c)(x_i - \mu_c)^T,$$

where $\mathcal{C}$ is the set of all the labels, $\mathcal{C}_c$ is the set of indices i for which $y_i = c \in \mathcal{C}$, N_c is the number of samples in $\mathcal{X}$ with class c, μ_c is the mean of the training samples in class c and μ is the mean of the whole training set. The solution to this problem can be found by taking the eigenvectors of $S_w^{-1} S_b$ associated with its largest eigenvalues to be the rows of L.

3.1.3 Average Neighborhood Margin Maximization (ANMM)

ANMM [36] is another DML technique specifically oriented to dimensionality reduction that tries to solve some of the limitations of PCA and LDA.

The objective of ANMM is to learn a linear transformation $L \in \mathcal{M}_{d' \times d}(\mathbb{R})$, with $d' \leq d$ that maximizes an *average neighborhood margin* defined, for each sample, by the difference between its average distance to its nearest neighbors of different class and the average distance to its nearest neighbors of same class.

If we consider the set $\mathcal{N}_i^o$ of the ξ samples in $\mathcal{X}$ nearest to x_i and with the same class as x_i, and the set $\mathcal{N}_i^e$ of the ζ samples in $\mathcal{X}$ nearest to x_i and with a different class to x_i, we can express the global average neighborhood margin, for the distance defined by L, as

$$\gamma^L = \sum_{i=1}^{N} \left(\sum_{k\,:\,x_k \in \mathcal{N}_i^e} \frac{\|Lx_i - Lx_k\|^2}{|\mathcal{N}_i^e|} - \sum_{j\,:\,x_j \in \mathcal{N}_i^o} \frac{\|Lx_i - Lx_j\|^2}{|\mathcal{N}_i^o|} \right).$$

In this expression, each summand is associated with each sample x_i in the training set, and the positive term inside each summand represents the average distance to its ζ nearest neighbors of different classes, while the negative term represents the average distance to its ξ nearest neighbors of the same class. Therefore, these differences constitute the average neighborhood margins for each sample x_i. The global margin γ^L can be expressed in terms of a scatterness matrix containing the information related to different-class neighbors, and a compactness matrix that stores the information corresponding to the same-class neighbors, that is,

$$\gamma^L = \operatorname{tr}(L(S - C)L^T),$$

where S and C are, respectively, the *scatterness* and *compactness* matrices, defined as

$$S = \sum_i \sum_{k\,:\,x_k \in \mathcal{N}_i^e} \frac{(x_i - x_k)(x_i - x_k)^T}{|\mathcal{N}_i^e|}$$

$$C = \sum_i \sum_{j\,:\,x_j \in \mathcal{N}_i^o} \frac{(x_i - x_j)(x_i - x_j)^T}{|\mathcal{N}_i^o|}.$$

If we impose the scaling restriction $LL^T = I$ (scaling would increase the average neighborhood margin indefinitely), the average neighborhood margin can be maximized by taking the eigenvectors of $S - C$ associated with its largest eigenvalues to be the rows of L.

3.2 Algorithms to Improve Nearest Neighbors Classifiers

One of the main applications of DML is to improve other distance based learning algorithms. Since the nearest neighbors classifier is one of the most popular distance based classifiers many DML algorithms are designed to improve this classifier, as is the case with LMNN [38] and NCA [39].

3.2.1 Large Margin Nearest Neighbors (LMNN)

LMNN [38] is a DML algorithm aimed specifically at improving the accuracy of the k-nearest neighbors classifier.

LMNN tries to bring each sample as close as possible to its *target neighbors*, which are k preselected same-class samples requested to become the nearest neighbors of the sample, while trying to prevent samples from other classes from invading a margin defined by those target neighbors. This setup allows the algorithm to locally separate the classes in an optimal way for k-neighbors classification.

Assuming the sets of target neighbors are chosen (usually they are taken as the nearest neighbors for Euclidean distance), the error function that LMNN minimizes is a two-term function. The first term is the target neighbors pulling term, given by

$$\varepsilon_{pull}(M) = \sum_{i=1}^{N} \sum_{j \rightsquigarrow i} d_M(x_i, x_j)^2,$$

where d_M is the Mahalanobis distance corresponding to $M \in S_d(\mathbb{R})_0^+$ and $j \rightsquigarrow i$ iff x_j is a target neighbor of x_i. The second term is the *impostors* pushing term, given by

$$\varepsilon_{push}(M) = \sum_{i=1}^{N} \sum_{j \rightsquigarrow i} \sum_{l=1}^{N} (1 - y_{il})[1 + d_M(x_i, x_j)^2 - d_M(x_i, x_l)^2]_+,$$

where $y_{il} = 1$ if $y_i = y_l$ and 0 otherwise, and $[\cdot]_+$ is defined as $[z]_+ = \max\{z, 0\}$. Finally, the objective function is given by

$$\varepsilon(M) = (1 - \mu)\varepsilon_{pull}(M) + \mu\varepsilon_{push}(M), \quad \mu \in]0, 1[.$$

This function can be optimized using semidefinite programming. It is possible to optimize this function in terms of L, using gradient methods, as well. By optimizing in terms of M we gain convexity in the problem, while by optimizing in terms of L we can use the algorithm to force a dimensionality reduction.

3.2.2 Neighborhood Components Analysis (NCA)

NCA [39] is another DML algorithm aimed specifically at improving the accuracy of the nearest neighbors classifiers. It is designed is to learn a linear transformation with the goal of minimizing the leave-one-out error expected by the nearest neighbor classification.

To do this, we define the probability that a sample $x_i \in \mathcal{X}$ has $x_j \in \mathcal{X}$ as its nearest neighbor for the distance defined by $L \in \mathcal{M}_d(\mathbb{R})$, p_{ij}^L, as the softmax

$$p_{ij}^L = \frac{\exp\left(-\|Lx_i - Lx_j\|^2\right)}{\sum\limits_{k \neq i} \exp\left(-\|Lx_i - Lx_k\|^2\right)} \ (j \neq i), \qquad p_{ii}^L = 0.$$

The expected number of correctly classified samples according to this probability is obtained as

$$f(L) = \sum_{i=1}^{N} \sum_{j \in C_i} p_{ij}^L,$$

where C_i is the set of indices j so that $y_j = y_i$. The function f can be maximized using gradient methods, and the distance resulting from this optimization is the one that minimizes the expected leave-one-out error, and therefore, the one that NCA learns.

3.3 Algorithms to Improve Nearest Centroids Classifiers

Apart from the nearest neighbors classifiers, other distance-based classifiers of interest are the so-called nearest centroid classifiers. These classifiers obtain a set of centroids for each class and classify a new sample by considering the nearest centroids to the sample. There are also DML algorithms designed for these classifiers, as is the case for NCMML and NCMC [40].

3.3.1 Nearest Class Mean Metric Learning (NCMML)

NCMML [40] is a DML algorithm specifically designed to improve the Nearest Class Mean (NCM) classifier. To do this, it uses a probabilistic approach similar to that used by NCA to improve the accuracy of the nearest neighbors classifier.

In this case, we define the probability that a sample $x_i \in \mathcal{X}$ will be labeled with the class c, according to the nearest class mean criterion, for the distance defined by $L \in \mathcal{M}_{d' \times d}(\mathbb{R})$, as

$$p_L(c|x) = \frac{\exp\left(-\frac{1}{2}\|L(x-\mu_c)\|^2\right)}{\sum_{c'\in\mathcal{C}} \exp\left(-\frac{1}{2}\|L(x-\mu_{c'})\|^2\right)},$$

where $\mathcal{C}$ is the set of all the classes and μ_c is the mean of the training samples with class c. The objective function that NCMML tries to maximize is the log-likelihood for the labeled data in the training set, according to the probability defined above, that is,

$$\mathcal{L}(L) = \frac{1}{N}\sum_{i=1}^{N} \log p_L(y_i|x_i).$$

This function can be optimized using gradient methods.

3.3.2 Nearest Class with Multiple Centroids (NCMC)

NCMC is the generalization of the nearest class mean classifier. In this classifier, a set with an arbitrary number of centroids is calculated for each class, using a clustering algorithm. Then, a new sample is classified by assigning the label of its nearest centroid.

An immediate generalization of NCMML allows us to learn a distance directed at improving NCMC. This DML algorithm is also referred to as NCMC. In this case, instead of the class means, we have a set of centroids $\{m_{c_j}\}_{j=1}^{k_c}$, for each class $c \in \mathcal{C}$. The generalized probability that a sample $x_i \in \mathcal{X}$ will be labeled with the class c is now given by $p_L(c|x) = \sum_{j=1}^{k_c} p_L(m_{c_j}|x)$, where $p_L(m_{c_j}|x)$ are the probabilities that m_{c_j} is the closest centroid to x, and is given by

$$p_L(m_{c_j}|x) = \frac{\exp\left(-\frac{1}{2}\|L(x-m_{c_j})\|^2\right)}{\sum_{c\in\mathcal{C}}\sum_{i=1}^{k_c} \exp\left(-\frac{1}{2}\|L(x-m_{c_i})\|^2\right)}.$$

Again, NCMC maximizes the log-likelihood function $\mathcal{L}(L) = \frac{1}{N}\sum_{i=1}^{N} p_L(y_i|x_i)$ using gradient methods.

3.4 Information Theory Based Algorithms

Several DML algorithms rely on information theory to learn their corresponding distances. The information theory concepts used in the algorithms we will introduce below are described in [48, Appendix B.3]. These algorithms have similar working schemes. First, they establish different probability distributions on the data, and then they try to bring these distributions closer or further away using divergences. The information theory based algorithms we will study are ITML [41], DMLMJ [42] and MCML [43].

3.4.1 Information Theoretic Metric Learning (ITML)

ITML [41] is a DML technique intended to find a distance metric as close as possible to an initial pre-defined distance, on which similarity and dissimilarity constraints for same-class and different-class samples are satisfied. This approach tries to preserve the properties of the original distance while adapting it to our dataset thanks to the restrictions it adds.

We will denote the positive definite matrix associated with the initial distance as M_0. Given any positive definite matrix $M \in S_d(\mathbb{R})^+$ and a fixed mean vector μ, we can construct a normal distribution $p(x|M)$ with mean μ and covariance M. ITML tries to minimize the Kullback-Leibler divergence between $p(x|M_0)$ and $p(x|M)$, subject to several similarity constraints on the data, that is

$$\begin{aligned} \min_{M \in S_d(\mathbb{R})^+} \quad & \mathrm{KL}(p(x|M_0) \| p(x|M)) \\ \text{s.t.:} \quad & d_M(x_i, x_j) \leq u, \quad (i,j) \in S \\ & d_M(x_i, x_j) \geq l, \quad (i,j) \in D, \end{aligned}$$

where S and D are sets of pairs of indices on the elements of $\mathcal{X}$ that represent the samples considered similar and not similar, respectively (normally, same-labeled pairs and different-labeled pairs), and u and l are, respectively, upper and lower bounds for the similarity and dissimilarity constraints. This problem can be optimized using gradient methods combined with iterated projections in order to fulfill the constraints.

3.4.2 Distance Metric Learning through the Maximization of the Jeffrey divergence (DMLMJ)

DMLMJ [42] is another DML technique based on information theory that tries to separate, with respect to the Jeffrey divergence, two probability distributions, the first associated with similar points while the second is associated with dissimilar points.

DMLMJ defines two difference spaces: a *k-positive difference space* that contains the differences between each sample in the dataset and its k-nearest neighbors from the same class, and a *k-negative difference space* that contains the differences between each sample and its k-nearest neighbors from different classes. Over these spaces, for a distance determined by a linear transformation $L \in \mathcal{M}_{d' \times d}(\mathbb{R})$, two gaussian distributions P_L and Q_L with equal mean are assumed. Then, the problem that DMLMJ optimizes is

$$\max_{L \in \mathcal{M}_{d' \times d}(\mathbb{R})} \quad f(L) = \mathrm{JF}(P_L \| Q_L) = \mathrm{KL}(P_L \| Q_L) + \mathrm{KL}(Q_L \| P_L).$$

This problem can be transformed into a trace optimization problem similar to those of PCA and LDA, and can also be solved by taking eigenvectors from the covariance matrices involved in the problem.

3.4.3 Maximally Collapsing Metric Learning (MCML)

MCML [43] is another DML technique based on information theory. The key idea of this algorithm is the fact that we would obtain an ideal class separation if we could project all the samples from the same class on a same point, far enough away from the points on which the rest of the classes would be projected.

In order to try to achieve this, MCML defines a probability that a sample x_j will be classified with the same label as x_i, with the distance given by a positive semidefinite matrix $M \in S_d(\mathbb{R})_0^+$, $p^M(j|i)$, as the softmax

$$p^M(j|i) = \frac{\exp(-d_M(x_i, x_j)^2)}{\sum_{k \neq i} \exp(-d_M(x_i, x_k)^2)}.$$

Then, it also defines a probability $p_0(j|i)$ for the ideal situation in which all the same-class samples collapse into the same point, far enough away from the collapsing points of the other classes, given by

$$p_0(j|i) \propto \begin{cases} 1, & y_i = y_j \\ 0, & y_i \neq y_j \end{cases}.$$

MCML tries to bring $p^M(\cdot|i)$ as close to the ideal $p^0(\cdot|i)$ as possible, for each i, using the Kullback-Leibler divergence between them. Therefore, the optimization problem is formulated as

$$\min_{M \in S_d(\mathbb{R})_0^+} \quad f(M) = \sum_{i=1}^{N} \mathrm{KL}\left[p_0(\cdot|i) \| p^M(\cdot|i)\right].$$

This function can be minimized using semidefinite programming.

3.5 Other Distance Metric Learning Techniques

In this section we will study some different proposals for DML techniques. The algorithms we will analyze are LSI [4], DML-eig [67] and LDML [68].

3.5.1 Learning with Side Information (LSI)

LSI [4], also sometimes referred to as Mahalanobis Metric for Clustering (MMC) is possibly one of the first algorithms that has helped make the concept of DML more well known. This algorithm is a global approach that tries to bring same-class data closer together while keeping data from different classes far enough apart.

Assuming that the sets S and D represent, respectively, pairs of samples that should be considered similar or dissimilar (i.e. samples that belong to the same class or to different classes, respectively), LSI looks for a positive semidefinite matrix $M \in S_d(\mathbb{R})_0^+$ that optimizes the following problem:

$$\begin{aligned} \min_{M} \quad & \sum_{(x_i,x_j)\in S} d_M(x_i,x_j)^2 \\ \text{s.t.:} \quad & \sum_{(x_i,x_j)\in D} d_M(x_i,x_j) \geq 1 \\ & M \in S_d(\mathbb{R})_0^+. \end{aligned}$$

This problem can be optimized using gradient descent together with iterated projections in order to fulfill the constraints.

3.5.2 Distance Metric Learning with eigenvalue optimization (DML-eig)

DML-eig [67] is a DML algorithm inspired by the LSI algorithm of the previous section, proposing a very similar optimization problem but offering a completely different resolution method, based on eigenvalue optimization.

We will once again consider the two sets S and D, of pairs of samples that are considered similar and dissimilar, respectively. DML-eig proposes an optimization problem that slightly differs from that proposed by the LSI algorithm, given by

$$\begin{aligned} \max_{M} \quad & \min_{(x_i,x_j)\in D} d_M(x_i,x_j)^2 \\ \text{s.t.:} \quad & \sum_{(x_i,x_j)\in S} d_M(x_i,x_j)^2 \leq 1 \\ & M \in S_d(\mathbb{R})_0^+. \end{aligned}$$

This problem can be transformed into a minimization problem for the largest eigenvalue of a symmetric matrix. This is a well-known problem and there are some iterative methods that allow this minimum to be reached [69].

3.5.3 Logistic Discriminant Metric Learning (LDML)

LDML [68] is a DML algorithm in which the optimization model makes use of the logistic function.

Recall that the *logistic* or *sigmoid* function is the map $\sigma : \mathbb{R} \to \mathbb{R}$ given by

$$\sigma(x) = \frac{1}{1+e^{-x}}.$$

In LDML, the logistic function is used to define a probability, which will assign the greater probability the smaller the distance between points. Given a positive semidefinite matrix $M \in S_d(\mathbb{R})_0^+$, this probability is expressed as

$$p_{ij,M} = \sigma(b - \|x_i - x_j\|_M^2),$$

where b is a positive threshold value that will determine the maximum value achievable by the logistic function, and that can be estimated by cross validation. LDML tries to maximize the log-likelihood given by

$$\mathcal{L}(M) = \sum_{i,j=1}^{N} y_{ij} \log p_{ij,M} + (1 - y_{ij}) \log(1 - p_{ij,M}),$$

where y_{ij} is a binary variable that takes the value 1 if $y_i = y_j$ and 0 otherwise. This function can be optimized using semidefinite programming.

3.6 Kernel Distance Metric Learning

Kernel methods constitute a paradigm within machine learning that is very useful in many of the problems addressed in this discipline. They usually arise in problems where the learning algorithm capability is reduced, typically due to the shape of the dataset. A classic learning algorithm where the kernel trick is very useful is the *Support Vector Machines (SVM)* classifier [70]. An example for this case is given in Figure 5.

In DML, the usefulness of kernel learning is a consequence of the limitations given by the Mahalanobis distances. Although learned metrics can later be used with non-linear classifiers, such as the nearest neighbors classifier, the metrics themselves are determined by linear transformations, which, in turn, are determined by the image of a basis in the departure space, which results in the fact that we only have the freedom to choose the image of as much data as the dimension has the space, mapping the rest of the vectors by linearity. When the amount of data is much larger than the space dimension this can become a limitation.

The kernel approach for DML follows a similar scheme to that of SVM. If we work with a dataset $\mathcal{X} = \{x_1, \dots, x_N\} \subset \mathbb{R}^d$, the idea is to send the data to a higher dimensional space, using a mapping $\phi : \mathbb{R}^d \to \mathcal{F}$, where $\mathcal{F}$ is a Hilbert space called the *feature space*, and then to learn in the feature space using a DML algorithm. The way we will learn a distance in the feature space will be via a continuous linear transformation $L : \mathcal{F} \to \mathbb{R}^{d'}$, where $d' \leq d$ (observe that L is not necessarily a matrix, since $\mathcal{F}$ is not necessarily finite dimensional), which we will also denote as $L \in \mathcal{L}(\mathcal{F}, \mathbb{R}^{d'})$.

As occurs with SVM, a great inconvenience arises when sending the data to the feature space, and that is that the problem dimension can greatly increase, and therefore the application of the algorithms can be very expensive computationally. In addition, if we want to work in infinite dimensional feature spaces, it is impossible to deal with the data in this case, unless we turn to the kernel trick.

We define the *kernel function* as the mapping $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ given by $K(x, x') = \langle \phi(x), \phi(x') \rangle$. The success of kernel functions is due to the fact that many learning algorithms only need to know the dot products between the elements in the training set to be able to work. This will happen in the DML algorithms we will study later. We can observe, as an example, that the calculation of Euclidean distances, which is essential in many DML

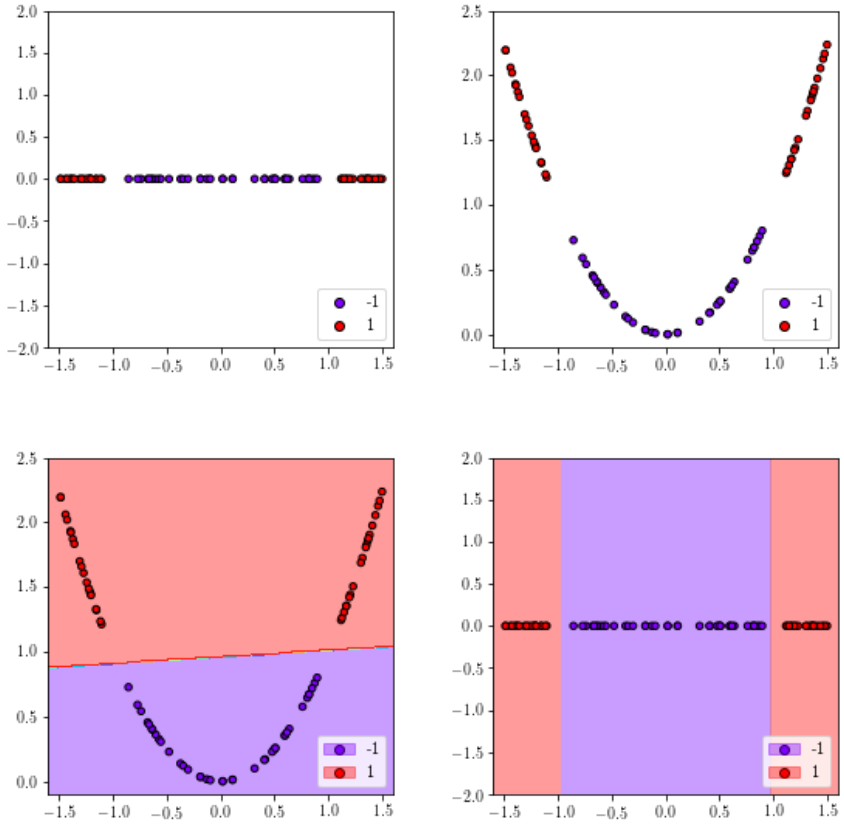


Figure 5: SVM and kernel trick. This binary classifier looks for the hyperplane that best separates both classes. Therefore, it is highly limited when the dataset is not separable by hyperplanes, as is the case for the dataset in the upper-left image. A solution consists of sending the data into a higher dimensional space, where data can be separated by hyperplanes, and apply the algorithm there, as it can be seen in the remaining images. The kernel trick allows us to execute the algorithm only in terms of the dot products of the samples in the new space, which makes it possible to work on very high dimensional spaces, or even infinite dimensional spaces. The existence of a *representer theorem* for SVM also allows the solution to be rewritten in terms of a vector with the size of the number of samples.

algorithms, can be made using only the kernel function. Indeed, for $x, x' \in \mathbb{R}^d$, we have

$$\begin{aligned}\|\phi(x) - \phi(x')\|^2 &= \langle \phi(x) - \phi(x'), \phi(x) - \phi(x') \rangle \\ &= \langle \phi(x), \phi(x) \rangle - 2\langle \phi(x), \phi(x') \rangle + \langle \phi(x'), \phi(x') \rangle \\ &= K(x,x) + K(x',x') - 2K(x,x').\end{aligned} \tag{1}$$

The next common problem for all the kernel-based DML algorithms is how to deal with the learned transformation. Since we are trying to learn a map $L \in \mathcal{L}(\mathcal{F}, \mathbb{R}^{d'})$, we may not be able to write it as a matrix, and when we can, this matrix may have dimensions that are too large. However, as L is continuous and linear, using the Riesz representation theorem, we can rewrite L as a vector of dot products by fixed vectors, that is, $L = (\langle \cdot, w_1 \rangle, \dots, \langle \cdot, w_{d'} \rangle)$, where $w_1, \dots, w_{d'} \in \mathcal{F}$. Furthermore, for the algorithms we will study, several *representer theorems* are known [71, 52, 45, 42, 72]. These theorems allow the vectors w_i to be expressed as a linear combination of the samples in the feature space, that is, for each $i \in \{1, \dots, d'\}$, there is a vector $\alpha^i = (\alpha^i_1, \dots, \alpha^i_N) \in \mathbb{R}^N$ so that $w_i = \sum_{j=1}^N \alpha^i_j \phi(x_j)$. Consequently, we can see that

$$L\phi(x) = A \begin{pmatrix} K(x_1, x) \\ \vdots \\ K(x_N, x) \end{pmatrix}, \tag{2}$$

where $A \in \mathcal{M}_{d' \times N}(\mathbb{R})$ is given by $A_{ij} = \alpha^i_j$.

Thanks to these theorems, we can address the problem computationally as long as we are able to calculate the coefficients of matrix A. When transforming a new sample it will suffice to construct the previous column matrix by evaluating the kernel function between the sample and each element in the training set, and then multiplying A by this matrix. On a final note, when training it is useful to view the kernel map as a matrix $K \in S_N(\mathbb{R})$, where $K_{ij} = K(x_i, x_j)$. A similar (in this case not necessarily square) matrix can be constructed when testing, with all the dot products between the train and test samples. By choosing the appropriate column of this matrix, we will be able to transform the corresponding test sample using Eq. 2.

Each DML technique that supports the use of kernels will use different tools for its performance, each one based on the original algorithms. In [48, Appendix C.6] we describe the kernelizations of some of the algorithms already introduced, namely, LMNN, ANMM, DMLMJ and LDA.

4 Experimental Framework and Results

With the algorithms introduced in the previous section, several experiments have been carried out. This section describes these experiments and shows the results.

4.1 Description of the Experiments

For the DML algorithms studied, a collection of experiments has been developed, consisting of the following procedures.

1. Evaluation of all the algorithms capable of learning at maximum dimension, applied to the k-NN classification, for different values of k.

2. Evaluation of the algorithms aimed at improving nearest centroid classifiers, applied to the corresponding centroid-based classifiers.

3. Evaluation of kernel-based algorithms, experimenting with different kernels, applied to the nearest neighbors classification.

4. Evaluation of algorithms capable of reducing dimensionality, for different dimensions, applied to the nearest neighbors classification.

When we mention in experiment 1 that an algorithm is "capable of learning at maximum dimension" we are excluding those dimensionality reduction algorithms that only learn a change of axes, as is the case with both PCA and ANMM, which at maximum dimension learn a transformation whose associated distance is still the Euclidean. LDA is kept, assuming that it will always take the maximum dimension that it is able to, which will be the number of classes of the problem. The algorithms directed at centroid-based classifiers are also excluded from experiment 1, together with those based on kernels, which will be analyzed in experiments 2 and 3, respectively.

The stated experiments indicate that the magnitude with which we will measure the performance of the algorithms is the result of the k-neighbors classification, except in the case of the algorithms based on centroids, which will use their corresponding classifier. These classifiers will be evaluated by a 10-fold cross validation. The results obtained from the predictions on the training set will also be included, in order to evaluate possible overfitting.

To evaluate the algorithms, we will use the implementations available in the Python library `pyDML` [47]. The algorithms will be executed using their default parameters, which can be found in the `pyDML` documentation[1]. These default parameters have been set with standard values. The following exceptions to the default parameters have been made:

- The LSI algorithm will have the parameter `supervised = True`, as it will be used for supervised learning.

- In the dimensionality reduction experiment (4), the algorithms will have the dimension number parameter set with the value of the dimension being evaluated.

- The parameter `k` of LMNN and KLMNN will be equal to the number of neighbors being considered in the nearest neighbors classification.

- LMNN will be executed with stochastic gradient descent, instead of semidefinite programming, in dimensionality reduction experiments, thus learning a linear transformation instead of a metric.

[1] `https://pydml.readthedocs.io/`

- The parameters `n_friends` and `n_enemies` of ANMM and KANMM will be equal to the number of neighbors being considered in the nearest neighbors classification.

- The parameter `n_neighbors` of DMLMJ and KDMLMJ will be equal to the number of neighbors being considered in the nearest neighbors classification.

- The parameter `centroids_num` of NCMC will be equal to the parameter `centroids_num` being considered in its corresponding classifier, `NCMC_Classifier`.

As for the datasets used in the experiments, up to 34 datasets have been collected and all of them are available in KEEL[2]. All these datasets are numeric, do not contain missing values, and are oriented to standard classification problems. In addition, although some of the DML algorithms scale well with the number of samples, others cannot deal with datasets that are too large, so it was decided that for sets with a high number of samples, a subset of a size that all algorithms can deal with, keeping the class distribution the same, would be selected. The characteristics of these datasets are described in Table 2. All datasets have been *min-max* normalized to the interval $[0, 1]$, feature to feature, prior to the execution of the experiments.

Finally, we describe the details of the experiments 1, 2, 3 and 4:

1. Algorithms will be evaluated with the classifiers 3-NN, 5-NN and 7-NN.

2. NCMML will be evaluated with the `Scikit-Learn` NCM classifier, while NCMC will be evaluated with its associated classifier, available in `pyDML`, for two different values: 2 centroids per class and 3 centroids per class.

3. Algorithms will be evaluated with 3-NN classifier, using the following kernels: linear (`Linear`), grade-2 (`Poly-2`) and grade-3 (`Poly-3`) polynomials, gaussian (`RBF`) and laplacian (`Laplacian`). The kernel version of PCA[3] will be also included in the comparison. Only the smallest datasets will be considered, so that they can be applicable to the algorithms that scale the worst with the dimension (recall that the kernel trick forces algorithms to work in dimensions of the order of the number of samples).

4. Algorithms will be evaluated with the classifiers 3-NN, 5-NN and 7-NN. The dimensions used are: $1, 2, 3, 5, 10, 20, 30, 40, 50$, the maximum dimension of the dataset, and the number of classes of the dataset minus 1. In this case, the following high-dimensionality datasets are selected: `sonar`, `movement_libras` and `spambase`. The algorithms to be evaluated in this experiment are: PCA, LDA, ANMM, DMLMJ, LMNN and NCA.

[2] KEEL, *knowledge extraction based on evolutionary learning* [73]: `http://www.keel.es/`.

[3] It is implemented in `Scikit-Learn`: `http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.KernelPCA.html`. Its theoretical details can be found in [72].

Dataset	Number of samples	Number of features	Number of classes
appendicitis	106	7	2
balance	625	4	3
bupa	345	6	2
cleveland	297	13	5
glass	214	9	7
hepatitis	80	19	2
ionosphere	351	33	2
iris	150	4	3
monk-2	432	6	2
newthyroid	215	5	3
sonar	208	60	2
wine	176	13	3
movement_libras	360	90	15
pima	768	8	2
vehicle	846	18	4
vowel	990	13	11
wdbc	569	30	2
wisconsin	683	9	2
banana (20 %)	1,060	2	2
digits	1,797	64	10
letter (10 %)	2,010	16	26
magic (10 %)	1,903	10	2
optdigits	1,127	64	10
page-blocks (20 %)	1,089	10	4
phoneme (20 %)	1,081	5	2
ring (20 %)	1,480	20	2
satimage (20 %)	1,289	36	7
segment (20 %)	462	19	7
spambase (10 %)	460	57	2
texture (20 %)	1,100	40	11
thyroid (20 %)	1,440	21	3
titanic	2,201	3	2
twonorm (20 %)	1,481	20	2
winequality-red	1,599	11	11

Table 2: Datasets used in the experiments.

4.2 Results

This section shows the results of the cross-validation for the different experiments. We will only show the results of the 3-NN classifier in the experiments that use nearest neighbors classifiers in this text. The results obtained for the remaining k-NN used in the experiments are available on the pyDML-Stats[4] website, where the results of all these experiments have been stored. The scripts used to do the experiments can also be found on this website. We have added the average score obtained and the average ranking to the results of the experiments 1, 2 and 3. The ranking has been made by assigning integer values between 1 and m, where m is the number of algorithms being compared in each experiment (adding half fractions in case of a tie), according to the position of the algorithms over each dataset, 1 being the best algorithm, and m the worst. The content of the different tables elaborated is described below.

- Table 3 shows the cross-validation results obtained for experiment 1, using the 3-NN score as the evaluation measure. Some cells do not show results because the algorithm did not converge.
- Table 4 shows the results of experiment 2. NCM and NCMC classifiers with 2 and 3 centroids per class were used as evaluation measures. For each classifier, the Euclidean distance (`Euclidean + CLF`) and the distance learning algorithm associated with the classifier (`NCMML / NCMC (2 ctrd) / NCMC (3 ctrd)`) have been evaluated.
- Table 5 shows the cross-validation results obtained on the training set for the kernel-based algorithms using the 3-NN classifier. Table 6 shows the corresponding results obtained on the test set.
- Table 7 shows the cross-validation results for experiment 4 in dataset `sonar`, using the classifier 3-NN. On the left are the results for the training set, and on the right, the results for the test set. Each row shows the results for the different dimensions evaluated. Tables 8 and 9 show the corresponding dimensionality results over the datasets `movement_libras` and `spambase`, respectively.

[4]Source code: `https://github.com/jlsuarezdiaz/pyDML-Stats`. The current website is located at `https://jlsuarezdiaz.github.io/software/pyDML/stats/versions/0.0.1-1/`

	Euclidean		LDA		ITML		DMLMJ		NCA		LMNN		LSI		DML-eig		MCML		LDML	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
appendicitis	.8428	.8339	.8428	.8522	.8533	.8604	.8490	.8256	**.8700**	.8504	.8407	.8422	.8659	**.8630**	.8585	.8622	.8502	.8513	.8669	.8422
balance	.8049	.8082	.8885	.8992	.8986	.8943	.8286	.8191	**.9592**	**.9584**	.8202	.8175	.9182	.9280	.8947	.8945	.8816	.8737	.8874	.8895
bupa	.6231	.6546	.6338	.6465	.6466	.6281	.6660	**.6776**	**.6943**	.5994	.6099	.6342	.6363	.6284	.5993	.6120	.5716	.5742	.5826	.5854
cleveland	.5570	.5468	.5694	.5502	.5488	.5523	.5626	.5636	**.6804**	.5436	.5780	.5803	.5518	.5722	.5896	.5829	.5978	.5785	.5784	**.5972**
glass	.6759	.7015	.6235	.6231	.6453	.6549	**.7092**	.7041	.7065	.6917	.6780	**.7067**	.6495	.6235	.6407	.6263	.6319	.5850	.6242	.6063
hepatitis	.8236	.8325	.9402	.8609	.9002	.8815	.8821	.8894	**.9569**	.8325	.9514	.8418	.9139	.9130	.9125	**.9176**	.9250	.8829	.9458	.8547
ionosphere	.8569	.8550	.8834	.8394	.8771	.8862	.8752	.8605	**.9534**	**.9084**	.9281	.8859	.8898	.8768	.8904	.8741	.9053	.8630	.8907	.8512
iris	.9533	.9533	.9681	.9533	.9703	.9733	.9585	.9666	.9755	.9666	.9481	.9400	.9703	**.9800**	.9585	.9600	.9688	.9466	**.9807**	.9600
monk-2	.9578	.9655	.9451	.9561	.9223	.9352	.9709	.9724	**1.000**	**1.000**	.9812	.9816	**1.000**	**1.000**	.9878	.9909	.9665	.9676	.9382	.9495
newthyroid	.9421	.9538	.9596	.9586	.9452	.9398	.9436	.9448	**.9700**	.9722	.9658	**.9725**	.9591	.9634	.9602	.9629	.9565	.9582	.9509	.9675
sonar	.8317	.8370	.9011	.7782	.8435	.8120	.9097	.8361	.9823	.8703	**.9941**	**.8742**	.8531	.8506	.8547	.7975	.8755	.8563	.8766	.7886
wine	.9606	.9606	.9968	**.9888**	.9900	.9773	.9812	.9662	.9956	.9882	.9956	.9832	.9837	.9662	.9975	.9767	**.9975**	.9832	.9956	**.9888**
movement_libras	.7972	.8139	**.8685**	.6642	.8038	.7992	.8460	**.8649**	.8516	.8319	.8065	.8020	.7351	.7440	.7970	.7872	.8063	.8073	.7256	.7360
pima	.7372	.7396	.7259	**.7525**	.7148	.7149	.7366	.7422	**.7841**	.7370	.7290	.7278	.7206	.7395	.7174	.7266	.7173	.7239	.7285	.7240
vehicle	.7077	.7125	.7698	**.7623**	.7625	.7516	.7643	.7551	**.8186**	.7550	.6855	.6757	.6590	.6666	.6506	.6501	.7398	.7369	.7186	.7170
vowel	.9699	.9787	.9680	.9777	.9423	.9535	.9751	**.9808**	**.9799**	**.9808**	.9693	.9777	.9436	.9474	.6719	.6757	.8558	.8737	.8885	.9090
wdbc	.9679	**.9716**	.9732	.9664	.9714	.9664	.9669	.9648	**.9751**	.9700	.9638	.9630	.9705	.9682	.9546	.9507	.9714	.9648	.9476	.9438
wisconsin	.9694	.9678	.9663	.9677	.9609	.9590	.9695	.9678	**.9723**	.9648	.9692	.9663	.9684	**.9722**	.9673	.9707	.9585	.9546	.9650	.9663
banana	.8543	.8555	.6504	.6469	.8536	.8556	.8550	.8565	.8553	**.8583**	**.8574**	.8583	.8535	.8517	.6718	.6878	.6282	.6102	.6268	.6319
digits	.9878	.9866	.9769	.9683	.9798	.9728	.9869	.9834	.9980	**.9894**	**.9993**	.9860	.9264	.9102	.8269	.8168	.9734	.9688	.9797	.9816
letter	.7174	.7208	.7955	.7967	.7161	.7195	.8163	.8204	**.8565**	**.8610**	.7048	.7162	.5396	.5496	.3191	.3214	.7600	.7534	.6217	.6372
magic	.8070	.8050	.7436	.7361	.8069	.8061	.8161	.8071	**.8396**	**.8145**	.7979	.7945	.7946	.7924	.7508	.7525	.7738	.7766	.7077	.6951
optdigits	.9756	.9777	.9671	.9512	.9731	.9669	.9770	.9761	.9956	.9759	**.9986**	**.9840**	.9398	.9306	.8164	.8022	.9761	.9591	.9596	.9591
page-blocks	.9495	.9495	**.9697**	**.9679**	.9614	.9614	.9515	.9504	.9637	.9577	.9459	.9439	-	-	.9515	.9523	.9613	.9642	.9438	.9404
phoneme	.7957	.7992	.7321	.7243	.7853	.7770	.7960	**.8002**	**.8044**	.7936	.7928	.7946	.7642	.7668	.7361	.7483	.7654	.7632	.7320	.7112
ring	.6410	.6432	.7289	.7101	.7290	.7352	.6440	.6453	**.9267**	**.8459**	.6750	.6615	.8331	.8162	.7308	.7223	.8315	.8223	.5634	.5648
satimage	.8585	.8564	.8541	.8387	.8495	.8341	.8670	**.8643**	**.8764**	.8511	.8565	.8558	.8490	.8465	.8153	.8130	.8246	.8171	.5427	.5501
segment	.8970	.9020	.9353	**.9370**	.9357	.9265	.9071	.9081	**.9451**	.9187	.9076	.8928	.8898	.8853	.9095	.9068	.9367	.9319	.8818	.8710
spambase	.8500	.8654	.9215	.8871	.8801	.8766	.8635	.8525	**.9391**	**.9154**	.9215	.9070	.9210	.9111	.9077	.9046	.9176	.9047	.9229	.8982
texture	.9560	.9618	**.9983**	**.9981**	.9801	.9754	.9864	.9854	.9843	.9800	.9180	.9218	.9333	.9400	.8979	.9009	.9740	.9745	.8658	.8718
thyroid	.9313	.9319	.9375	.9450	.9397	.9402	.9355	.9361	.9459	.9395	.9320	.9319	.9357	.9354	.9458	.9485	.9377	.9320	**.9587**	**.9583**
titanic	.7607	.7583	**.7727**	**.7804**	.7682	.7609	.7612	.7587	.5709	.6764	.6018	.6964	-	-	.7107	.7331	.7150	.7253	.7108	.7341
twonorm	.9609	.9595	.9778	.9750	.9685	.9669	.9612	.9561	**.9817**	.9790	.9778	.9756	.9776	.9770	.9782	**.9810**	.9708	.9730	.9789	.9804
winequality-red	.5808	**.5865**	.5657	.5733	.5754	.5828	.5828	.5860	**.6022**	.5766	.5647	.5772	.5656	.5809	.5281	.5292	.5675	.5611	.5376	.5471
AVG RANKING	6.558	5.661	5.147	5.426	5.808	5.382	4.926	4.544	**1.705**	**3.661**	5.220	5.279	6.411	5.382	6.691	6.220	5.735	6.279	6.794	7.161
AVG SCORE	.8383	.8425	.8515	.8363	.8500	.8470	.8560	.8526	**.8886**	**.8634**	.8490	.8432	.8410	.8405	.8059	.8041	.8438	.8359	.8125	.8062

Table 3: Results of cross-validation with 3-NN.

	Euclidean + NCM		NCMML		Euclidean + NCM (2 ctrd)		NCMC (2 ctrd)		Euclidean + NCM (3 ctrd)		NCMC (3 ctrd)	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
appendicitis	.8365	**.8640**	**.8512**	.8440	.6479	.6031	.6248	.6246	.8102	.7800	.7557	.7266
balance	.7496	.7475	.6865	.6864	.7004	.6721	.8280	**.8225**	.7160	.6654	**.8321**	.8222
bupa	.5996	.6004	**.6628**	**.6407**	.6270	.6058	.6247	.6260	.6589	.5903	.6409	.5826
cleveland	.5742	.5377	.6296	**.5516**	.5727	.5050	.6280	.5093	.6203	.4871	**.6435**	.5135
glass	.5218	.4862	.6221	.5290	.6479	.5849	.5877	.5372	.6427	.5208	**.7461**	**.6421**
hepatitis	.8500	.8293	.9832	**.8688**	.8792	.8436	.9513	.8373	.8986	.7946	**.9833**	.8672
ionosphere	.7445	.7378	.9313	.8797	.9186	**.8889**	.9018	.8764	.9062	.8750	**.9477**	.8886
iris	.9318	.9133	**.9814**	.9600	.9696	**.9666**	.9711	.9600	.9696	.9466	.9800	.9600
monk-2	.8078	.8104	.7950	.7923	.8071	.8082	.8112	.8038	.8127	.7895	**.8457**	**.8237**
newthyroid	.9359	.9352	**.9798**	.9634	.9498	.9448	.9741	**.9725**	.9695	.9681	.9757	.9629
sonar	.7265	.7017	.9257	.7641	.7120	.6929	.9219	.7839	.8360	.7437	**.9412**	**.7982**
wine	.9687	.9495	**1.000**	.9663	.9825	.9659	.9918	**.9826**	.9762	.9432	.9912	.9704
movement_libras	.6358	.5946	.8176	.7575	.7777	.6764	.8803	.7852	.8717	.7749	**.9420**	**.8373**
pima	.7337	.7279	**.7717**	**.7604**	.7565	.7382	.6720	.6641	.7521	.7461	.7322	.7253
vehicle	.4545	.4491	**.7998**	**.7797**	.6066	.5824	.7429	.7219	.6537	.6179	.7558	.7244
vowel	.5367	.5070	.6689	.6383	.6087	.5717	.7272	.6868	.5732	.5333	**.7690**	**.7292**
wdbc	.9388	.9367	.9794	.9649	.9548	.9385	.9796	**.9736**	.9703	.9665	**.9810**	.9701
wisconsin	.9648	.9648	**.9681**	**.9662**	.9586	.9502	.9655	.9603	.9515	.9486	.9541	.9487
banana	.5769	.5737	.5571	.5558	.6417	.6359	.5846	.5859	.7516	.7573	**.7828**	**.7766**
digits	.9066	.8981	.8047	.8083	.9521	.9415	.8664	.8586	**.9723**	**.9644**	**.8893**	.8554
letter	.5707	.5341	.7114	.6868	.5895	.5282	.7251	.6902	.6601	.5714	**.7825**	**.7301**
magic	.7712	.7693	.7790	.7751	.7786	.7745	**.7947**	**.7861**	.7600	.7509	.7820	.7751
optdigits	.9173	.9104	.8018	.7978	.9479	.9352	.8542	.8185	**.9675**	**.9609**	.8923	.8641
page-blocks	.8133	.8165	**.9636**	**.9576**	.8219	.8221	.9090	.9071	.8680	.8696	.8966	.8953
phoneme	.7417	.7399	.7587	.7538	**.7683**	**.7667**	.7084	.7159	.7207	.7149	.7091	.7048
ring	.7780	.7723	.7819	**.7784**	.8002	.7601	.7197	.7012	**.8160**	.7750	.6844	.6636
satimage	.7868	.7844	**.8478**	**.8255**	.8052	.7921	.8342	.8123	.8244	.7844	.8362	.8007
segment	.8460	.8367	**.9437**	**.9037**	.8596	.8452	.9203	.8976	.8581	.8030	.9242	.8874
spambase	.8874	.8827	**.9584**	.9154	.8816	.8763	.9400	**.9241**	.8985	.8893	.9335	.9112
texture	.7445	.7372	**.9912**	**.9781**	.8586	.8500	.9694	.9581	.9079	.8900	.9759	.9654
thyroid	.4532	.4394	**.8163**	**.8082**	.5724	.5558	.6875	.6922	.5959	.5630	.7469	.7358
titanic	.7540	.7459	**.7825**	**.7854**	.6746	.6516	.5624	.6550	.5630	.6824	.7279	.7350
twonorm	.9807	**.9824**	.9854	.9797	.9799	.9723	.9847	.9790	.9787	.9743	**.9855**	.9777
winequality-red	.3519	.3371	**.4535**	**.4359**	.4067	.3838	.4031	.3870	.3940	.3582	.4024	.3738
AVG RANKING	4.941	4.441	**2.382**	**2.500**	4.117	4.000	3.411	2.911	3.764	4.235	**2.382**	2.911
AVG SCORE	.7468	.7369	.8233	.7958	.7770	.7538	.8014	.7793	.7978	.7647	**.8344**	**.7984**

Table 4: Results of the experiments with NCMML and NCMC.

	EUC	KPCA					KDA					KANMM					KDMLMJ					KLMNN				
		Lin.	Poly-2	Poly-3	RBF	Lapl.	Lin.	Poly-2	Poly-3	RBF	Lapl.	Lin.	Poly-2	Poly-3	RBF	Lapl.	Lin.	Poly-2	Poly-3	RBF	Lapl.	Lin.	Poly-2	Poly-3	RBF	Lapl.
appendicitis	.8428	.8428	.8407	.8355	.8428	.8438	.8481	.8491	.8502	.8679	.8564	.8543	.8543	.8523	.8763	.8889	.8491	.8397	.8407	.8575	.8805	.8355	.8397	.8355	.8523	**1.000**
balance	.8049	.8217	.7770	.7923	.8136	.8288	.4547	.6756	.6279	.7497	.8838	.7764	.7980	.8224	.8462	.8327	.8298	.9509	.9696	.9374	.9491	.8202	.8426	.9079	.8865	**.9852**
bupa	.6231	.6231	.6235	.6222	.6231	.6560	.5623	.5677	.5758	.5526	.5665	.5903	.5890	.5832	.5639	.5735	.6586	.6254	.6457	.6247	.7075	.6437	.6547	.6627	.6721	**.9987**
cleveland	.5570	.5570	.5537	.5552	.5570	.5417	.5301	.5424	.5480	.5585	.5394	.5664	.5705	.5698	.5559	.5435	.5615	.5514	.5544	.5473	.5499	.5536	.5862	.6389	.6753	**.9943**
glass	.6759	.6759	.6801	.6801	.6759	.6931	.6376	.6723	.6459	.6474	.6661	.6884	.6853	.6874	.6724	.6848	.6977	.6957	.7024	.6702	.7342	.6790	.7200	.7289	.7372	**.9890**
hepatitis	.8236	.8236	.8083	.8111	.8236	.8403	.8318	.8416	.8471	.8181	.8111	.8695	.8709	.8653	.8363	.8306	.8834	.8708	.9179	.8708	.9208	.9777	**1.000**	**1.000**	**1.000**	**1.000**
ionosphere	.8569	.8569	.8518	.8493	.8569	.8882	.6986	.6749	.6860	.7641	.7429	.8512	.8518	.8502	.9259	.9449	.8752	.8714	.8774	.8819	.9398	.8695	.9670	.9692	.9838	**1.000**
iris	.9533	.9533	.9540	.9548	.9533	.9518	.8555	.9429	.9488	.9562	.9362	.9459	.9503	.9518	.9451	.9466	.9592	.9503	.9555	.9474	.9592	.9651	.9681	.9659	.9681	**1.000**
monk-2	.9578	.9503	.9603	.9580	.9480	.9511	.7047	.7119	.6995	.8320	.8459	.6877	.6365	.6252	.8996	.9964	.9709	.9763	.9863	.9588	.9938	.9789	.9873	.9920	.9989	**1.000**
newthyroid	.9421	.9421	.9395	.9385	.9421	.9488	.9581	.9576	.9571	.9638	.9612	.9390	.9400	.9410	.9617	.9643	.9441	.9478	.9421	.9503	.9612	.9690	.9700	.9731	.9741	**1.000**
sonar	.8317	.8317	.8365	.8370	.8317	.8392	.6261	.6287	.6410	.6057	.6832	.7499	.7516	.7521	.8263	.8450	.9076	.8696	.8637	.8691	.9332	.9706	**1.000**	**1.000**	**1.000**	**1.000**
wine	.9606	.9606	.9625	.9606	.9606	.9669	.9269	.9169	.9200	.9644	.9694	.9325	.9332	.9325	.9844	.9825	.9825	.9793	.9831	.9819	.9956	.9993	**1.000**	**1.000**	**1.000**	**1.000**
movement_libras	.7972	.7972	.7866	.7805	.7972	.7775	.3322	.4953	.4980	.7419	.7544	.4684	.4684	.4689	.7195	.7376	.8590	.8118	.8272	.8124	.8450	.7945	.8776	.8924	.9246	**1.000**
pima	.7372	.7372	.7361	.7377	.7372	.7181	.6820	.6753	.6653	.6535	.6650	.7170	.7144	.7141	.7164	.7076	.7359	.7381	.7460	.7455	.7628	.7378	.7453	.7397	.7505	**.9968**
banana	.8543	.8546	.8549	.8562	.8545	.8536	.6730	.6622	.6672	.7037	.6493	.8593	.8603	.8624	.7854	.8167	.8554	.8545	.8551	.8423	.7643	.8546	.8546	.8547	.8540	**.9801**
optdigits	.9756	.9756	.9761	.9753	.9756	.9664	.9186	.9190	.9198	.9434	.9386	.9379	.9387	.9394	.9558	.9543	.9767	.9790	.9813	.9791	.9857	.9905	.9999	.9997	.9998	**1.000**
phoneme	.7957	.7957	.7940	.7935	.7957	.7962	.6960	.7019	.6929	.7112	.7271	.7727	.7726	.7732	.7735	.7880	.7959	.7940	.7945	.7898	.7858	.8021	.7963	.7850	.7994	**.9829**
satimage	.8585	.8585	.8583	.8594	.8585	.8641	.8104	.8114	.8097	.8422	.8452	.8188	.8186	.8190	.8535	.8601	.8707	.8623	.8622	.8590	.8680	.8623	.8682	.8682	.8803	**.9935**
segment	.8970	.8970	.8972	.8968	.8970	.9004	.8360	.8278	.8276	.8187	.8737	.8309	.8300	.8300	.8323	.8723	.9049	.8965	.8994	.8879	.9343	.9244	.9383	.9405	.9415	**.9985**
spambase	.8500	.8500	.8500	.8504	.8500	.8328	.8490	.8323	.8313	.7263	.7500	.8777	.8775	.8777	.8635	.8988	.8642	.8850	.8828	.8830	.9014	.9369	.9495	.9459	.9497	**.9990**
twonorm	.9609	.9609	.9636	.9648	.9609	.9531	.9765	.9758	.9758	.9765	.9756	.9798	.9800	.9804	.9763	.9751	.9616	.9642	.9624	.9672	.9810	.9773	.9847	.9800	.9861	**.9915**
AVG RANKING	15.45	15.26	15.52	15.95	15.47	14.14	22.19	21.64	21.50	18.95	19.47	17.14	16.61	16.30	15.52	13.66	9.690	11.71	9.880	12.33	7.023	9.476	5.380	5.833	3.619	**1.214**
AVG SCORE	.8360	.8365	.8336	.8338	.8360	.8387	.7337	.7563	.7541	.7808	.7924	.7959	.7949	.7952	.8271	.8402	.8545	.8530	.8595	.8506	.8740	.8639	.8833	.8895	.8969	**.9957**

Table 5: Results of kernel experiments on the training set.

	EUC	KPCA					KDA					KANMM					KDMLMJ					KLMNN				
		Lin.	Poly-2	Poly-3	RBF	Lapl.	Lin.	Poly-2	Poly-3	RBF	Lapl.	Lin.	Poly-2	Poly-3	RBF	Lapl.	Lin.	Poly-2	Poly-3	RBF	Lapl.	Lin.	Poly-2	Poly-3	RBF	Lapl.
appendicitis	.8339	.8339	.8339	.8248	.8339	.8546	.8613	.8813	.8813	.8713	.8622	.8646	.8446	.8446	.8813	**.8904**	.8339	.8248	.8339	.8339	.8248	.8322	.8248	.8331	.8057	.8422
balance	.8082	.8383	.7823	.8032	.8428	.8336	.4597	.6699	.6150	.7407	.8899	.7738	.8058	.8382	.8559	.8320	.8144	.9582	**.9711**	.9374	.9519	.8222	.8367	.9118	.8736	.8464
bupa	.6546	.6546	.6661	.6662	.6546	.6862	.5620	.5191	.5566	.5244	.5389	.5963	.6022	.5908	.6199	.5794	.6777	.6375	.6371	.6310	**.6924**	.6574	.6402	.6516	.6632	.6467
cleveland	.5468	.5468	.5432	.5535	.5468	.5528	.5541	.5449	.5485	**.5860**	.5474	.5689	.5653	.5685	.5657	.5623	.5605	.5682	.5774	.5674	.5688	.5736	.5744	.5628	.5184	.5325
glass	.7015	.7015	.7102	.7102	.7015	.7117	.5971	.6708	.6543	.6582	.6630	.6777	.6677	.6677	.7026	.6890	.7020	.6910	.6910	.6715	**.7351**	.6907	.6911	.6827	.6845	.7334
hepatitis	.8325	.8325	.8343	.8343	.8325	.8436	.8123	.8140	.8265	.8325	.8436	.8732	.8732	.8732	.8575	.8575	**.8894**	.8672	.8547	.8547	.8644	.8408	.8672	.8404	.8672	.8845
ionosphere	.8550	.8550	.8520	.8491	.8550	.8885	.7119	.6695	.6977	.7638	.7598	.8517	.8489	.8461	.9258	**.9457**	.8634	.8607	.8606	.8666	.9316	.8463	.9081	.8910	.8969	.9396
iris	.9533	.9533	.9533	.9533	.9533	.9533	.8800	.9533	.9533	**.9800**	.9333	.9600	.9600	.9533	.9533	.9466	.9600	.9600	.9533	.9533	.9533	.9533	.9533	.9600	.9466	.9266
monk-2	.9655	.9539	.9677	.9655	.9634	.9585	.7294	.7203	.6880	.8176	.8385	.6877	.6529	.6435	.9213	.9930	.9724	.9699	.9862	.9654	.9908	.9817	.9863	.9817	.9863	**1.000**
newthyroid	.9538	.9538	.9448	.9448	.9538	.9493	.9538	.9538	.9538	.9586	.9491	.9396	.9396	.9396	.9580	.9627	.9493	.9493	.9448	.9491	.9489	.9584	.9632	**.9679**	.9632	.9625
sonar	.8370	.8370	.8515	.8515	.8370	.8560	.5812	.6540	.6431	.5948	.6872	.7451	.7451	.7451	.8225	.8267	.8461	.8556	.8560	.8651	**.8753**	.7653	.8701	.8408	.8704	.8316
wine	.9606	.9606	.9606	.9606	.9606	.9613	.9214	.9047	.9158	.9367	.9603	.9262	.9318	.9318	.9888	.9835	.9606	.9780	.9724	.9777	.9780	**.9888**	.9830	.9666	**.9888**	.9777
movement_libras	.8139	.8139	.8070	.7948	.8139	.8059	.3715	.5209	.5187	.7631	.7600	.4969	.5006	.4982	.7615	.7566	**.8718**	.8251	.8406	.8219	.8234	.8106	.8315	.8093	.8375	.7989
pima	.7396	.7396	.7370	.7318	.7396	.7175	.6967	.6850	.7150	.6810	.6784	.7238	.7251	.7278	.7134	.7109	.7383	.7396	**.7513**	.7487	.7408	.7462	.7461	.7370	.7370	.7005
banana	.8555	.8555	.8546	.8546	.8555	.8574	.6688	.6642	.6934	.7030	.6169	.8583	.8592	**.8611**	.7810	.8177	.8565	.8546	.8536	.8425	.7622	.8536	.8508	.8565	.8565	.8414
optdigits	.9777	.9777	.9795	.9777	.9777	.9706	.9146	.9155	.9164	.9419	.9350	.9359	.9359	.9377	.9565	.9529	.9752	**.9804**	.9787	.9777	.9804	.9607	.9760	.9697	.9671	.9572
phoneme	.7992	.7992	.7964	.7973	.7992	.8002	.6847	.7068	.7067	.7132	.7270	.7826	.7845	.7854	.7724	.7880	.8030	.8029	.7964	.7918	.7845	**.8047**	.8001	.7854	.7991	.7835
satimage	.8564	.8564	.8564	.8580	.8564	.8612	.8053	.8138	.8122	.8364	.8496	.8193	.8185	.8170	.8535	.8589	**.8689**	.8589	.8580	.8527	.8559	.8473	.8565	.8487	.8425	.8558
segment	.9020	.9020	.9020	.9000	.9020	.9000	.8404	.8363	.8323	.8112	.8625	.8346	.8346	.8326	.8227	.8687	.9098	.8768	.8840	.8738	.9183	.9153	.9241	.9224	.9285	.9517
spambase	.8654	.8654	.8676	.8676	.8632	.8110	.8372	.8348	.8282	.7151	.7303	.8807	.8807	.8807	.8567	.8849	.8546	.8740	.8784	.8826	.8827	.8914	.9027	.8981	**.9049**	.8999
twonorm	.9595	.9595	.9649	.9642	.9595	.9567	.9804	.9777	.9770	.9797	.9730	**.9810**	.9797	.9790	.9743	.9682	.9561	.9635	.9574	.9675	.9702	.9689	.9702	.9682	.9655	.9635
AVG RANKING	12.97	12.83	13.21	13.57	12.76	11.73	21.42	20.73	20.64	17.92	19.21	15.57	15.95	16.16	13.40	12.23	9.738	9.642	10.21	11.61	8.404	11.02	**7.880**	10.38	10.02	11.69
AVG SCORE	.8415	.8424	.8412	.8411	.8430	.8443	.7345	.7577	.7588	.7814	.7908	.7990	.7979	.7982	.8354	.8417	.8507	.8522	.8541	.8492	**.8588**	.8433	.8551	.8517	.8525	.8512

Table 6: Results of kernel experiments on the test set.

	PCA	LDA	ANMM	DMLMJ	NCA	LMNN
1	.5016	.9011	.6965	.7826	.9214	.7237
2	.5891	-	.7670	.8050	.9807	.8782
3	.7729	-	.8359	.8333	.9770	.9513
5	.8215	-	.8904	.9033	.9759	.9914
10	.8600	-	.8958	.9652	.9764	.9994
20	.8541	-	.8872	.9583	.9668	**1.000**
30	.8456	-	.8627	.9508	.9706	**1.000**
40	.8365	-	.8424	.9460	.9839	**1.000**
50	.8312	-	.8370	.9263	.9850	**1.000**
Max. Dimension	.8317	-	.8317	.9097	.9823	**1.000**
N. Classes - 1	.5016	.9011	.6965	.7826	.9214	.7237

	PCA	LDA	ANMM	DMLMJ	NCA	LMNN
1	.5619	.7782	.6770	.7256	.8073	.6640
2	.6293	-	.7541	.7113	.8077	.7593
3	.7641	-	.8395	.7741	.8265	.8077
5	.8075	-	.8263	.8182	.8220	.8408
10	.8699	-	.8751	.8651	.8270	.8654
20	.8601	-	.8749	**.8844**	.8699	.8703
30	.8610	-	.8649	.8749	.8653	.8754
40	.8465	-	.8610	.8697	.8792	.8613
50	.8565	-	.8515	.8654	.8558	.8706
Max. Dimension	.8370	-	.8370	.8361	.8703	.8706
N. Classes - 1	.5619	.7782	.6770	.7256	.8073	.6640

Table 7: Results of dimensionality reduction experiments on `sonar` with 3-NN (train - test)

	PCA	LDA	ANMM	DMLMJ	NCA	LMNN
1	.1938	.3339	.2414	.2720	.3360	.2547
2	.2813	.5362	.4597	.4720	.6638	.5416
3	.5232	.6143	.6435	.6684	.7195	.6900
5	.6873	.7211	.7473	.7918	.8188	.8156
10	.7831	.8661	.8053	.8857	.8383	.8485
20	.7978	-	.7972	.8705	.8442	.8490
30	.7981	-	.7978	.8652	.8438	.8514
40	.7972	-	.7975	.8594	.8469	.8526
50	.7972	-	.7972	.8538	.8431	.8498
Max. Dimension	.7972	-	.7972	.8460	.8516	.8490
N. Classes - 1	.7932	.8685	.8061	**.8901**	.8398	.8438

	PCA	LDA	ANMM	DMLMJ	NCA	LMNN
1	.1747	.3169	.2675	.2694	.2606	.2673
2	.2574	.4553	.4800	.4476	.6181	.5251
3	.5483	.4978	.6680	.6684	.6920	.6499
5	.7177	.5938	.7763	.7774	.7655	.8012
10	.8007	.7001	.8119	.8711	.8017	.8220
20	.8139	-	.8106	**.8829**	.8143	.8333
30	.8139	-	.8139	.8696	.8191	.8133
40	.8139	-	.8139	.8605	.8323	.8233
50	.8139	-	.8139	.8627	.8310	.8255
Max. Dimension	.8139	-	.8139	.8649	.8319	.8133
N. Classes - 1	.8185	.6642	.8137	.8811	.8274	.8211

Table 8: Results of dimensionality reduction experiments on `movement_libras` with 3-NN (train - test)

	PCA	LDA	ANMM	DMLMJ	NCA	LMNN
1	.8369	.9215	.8567	.6995	**.9420**	.9340
2	.8316	-	.8869	.7724	**.9420**	.9386
3	.8487	-	.8973	.8886	.9388	.9335
5	.8784	-	.9079	.9009	.9415	.9335
10	.8681	-	.9222	.9195	.9400	.9318
20	.8700	-	.9067	.9217	.9400	.9297
30	.8586	-	.8787	.8867	.9403	.9328
40	.8572	-	.8654	.8727	.9369	.9318
50	.8536	-	.8560	.8596	.9374	.9299
Max. Dimension	.8500	-	.8500	.8635	.9391	.9285
N. Classes - 1	.8369	.9215	.8567	.6995	**.9420**	.9340

	PCA	LDA	ANMM	DMLMJ	NCA	LMNN
1	.8106	.8871	.8587	.6588	.9044	.8872
2	.8261	-	.8850	.7173	**.9197**	.8958
3	.8543	-	.9090	.8807	.9152	.9068
5	.8782	-	.9049	.8700	.9111	.9069
10	.8826	-	.9198	.9044	.9153	.9113
20	.8695	-	.9048	.8937	.9155	.9005
30	.8502	-	.8675	.8851	.9154	.9027
40	.8547	-	.8567	.8611	.9111	.9005
50	.8655	-	.8633	.8569	.9133	.9070
Max. Dimension	.8654	-	.8654	.8525	.9154	.9092
N. Classes - 1	.8106	.8871	.8587	.6588	.9044	.8872

Table 9: Results of dimensionality reduction experiments on `spambase` with 3-NN (train - test)

4.3 Analysis of Results

4.3.1 In-depth analysis

Below we will describe the main details observed in the algorithms for the different experiments carried out.

- **NCA.** In terms of the results obtained in the first experiment, we can clearly see that NCA has obtained the best results. This is partly due to the fact that the algorithms have been evaluated with nearest neighbors classifiers, and that NCA was specifically designed to improve this classifier. NCA came first in most of the validations over the training set, showing its ability to fit to the data, but it has also obtained clear victories in many of the datasets over the test set, thus also demonstrating a great capacity for generalization. We have to note that NCA (and also other algorithms such as LMNN) commits substantial errors in datasets such as *titanic* [73]. This is a numerical-transformed dataset, but of a categorical nature, and with many repeated elements that may belong to different classes. This may be causing highly discriminative algorithms such as NCA or LMNN not being able to transform the dataset appropriately. This justifies how in certain situations other algorithms can be more useful than those that show better behavior in general [74].

- **LMNN and DMLMJ.** We can also see that DMLMJ and LMNN algorithms stand out, although not as much as NCA. These algorithms are also directed at nearest neighbor classification, which justifies these good results. LMNN seems to have a slow convergence with the projected gradient method, and it could have achieved better results with a greater number of iterations. In fact, in the analysis of dimensionality reduction experiments we will observe that LMNN performs much better with the stochastic gradient descent method.

- **LSI.** LSI is another algorithm that is capable of obtaining very good results on certain datasets, but it is penalized by many others, where it is not able to optimize enough, not even being able to converge in several datasets.

- **ITML and MCML.** ITML and MCML are two algorithms that, despite getting the best results in a very small number of cases, they get decent results in most datasets, resulting in quite a stable performance. ITML does not learn too much from the characteristics of the training set, but is able to generalize what has been learned in quite an effective way, being possibly the algorithm that loses the least accuracy over the test set, with respect to the training set. On the other hand, MCML has more learning capacity, even showing a slight overfitting, as its results are worse than those of many algorithms on the test set.

- **LDA.** Another algorithm in which we can see overfitting, perhaps more clearly, is LDA. This algorithm is capable of getting very good results on the training set, surpassing most of the algorithms, but it gets noticeably worse when evaluated on

the test dataset. Recall that LDA is able to learn only a maximum dimension equal to the number of classes of the dataset minus one. This may be causing a loss of important information on many datasets by the projection it learns.

- **DML-eig and LDML.** Finally, although DML-eig and LDML are able to get better results than Euclidean distance on the training sets, on several datasets they have obtained quite low quality results. On many of the test datasets, they are surpassed by the Euclidean distance.

- **Untrained k-NN**. The untrained k-NN or, equivalently, the classical k-NN with Euclidean distance, is always outperformed by some of the distance-learned k-NNs in the training set, and is also mostly outperformed in the test set. This shows the benefits of learning a distance as opposed to the traditional use of the nearest neighbor classifier. The untrained k-NN also shows better average results on the test set than on the training set, and is the only one among all the compared algorithms. This may be due to the fact that, as it is not using a pretrained distance, it is unlikely to overfit, although according to the results there is a lot of room for improvement in both training and test sets for this basic version.

- **NCMML and NCMC.** If we analyze the results of the centroid-based classifiers, we can easily observe that in the vast majority of cases the classifier has worked much better after learning the distance with its associated learning algorithm, than it has by using the Euclidean distance. It can also be observed that the results are subject to great variability, depending on the number of centroids chosen. This shows that the choice of an adequate number of centroids that adapt well to the disposition of the different classes is fundamental to achieve successful learning with these algorithms.

- **Kernel algorithms.** Focusing now on the kernel-based algorithms, it is interesting to note how KLMNN with laplacian kernel is able to adjust as much as possible to the data, getting a 100 % success rate on most of the datasets. This success rate is not transferred, in general, to the test data, showing that this algorithm overfits with laplacian kernel. We can also observe that the best results are distributed in a varied way among the different evaluated options. The choice of a suitable kernel that fits well with the disposition of the data is decisive for the performance of kernel-based algorithms.

- **Dimensionality reduction experiments.** To conclude our analysis, dimensionality experiments allow us to observe that the best results are not always obtained when considering the maximum dimension. This may be due to the fact that the algorithms are able to denoise the data, ensuring that the classifier used later does not overfit. We also see that we cannot reduce the dimension as much as we want, because at some point we start losing information, which happens in many cases with LDA, which is its great limitation. In general, we can observe that all algorithms improve their results by reducing dimensionality until a certain value, al-

though the best results are provided by LMNN, DMLMJ and NCA. The results obtained by LMNN open the possibility of using this algorithm with stochastic gradient descent, instead of the semidefinite programming algorithm used in the first experiment, since the results it provides are quite good. Although these algorithms have obtained better results, the use of ANMM and LDA (as long as the dimension allows it) is important for the estimation of an adequate dimension, since they are much more efficient than the first ones. As for PCA, it gets the worst results in low dimensions, probably due to not considering the information of the labels.

4.3.2 Global analysis

In order to complete the verbal analysis, we have developed a series of Bayesian statistical tests to assess the extent to which the performance of the different algorithms analyzed outperforms the other algorithms. To do this, we have elaborated several pairwise Bayesian sign tests [46]. In these tests, we will consider the differences between the obtained scores of two algorithms, assuming that their prior distribution is a Dirichlet Process [75], defined by a prior strength $s = 1$ and a prior pseudo-observation $z_0 = 0$. After considering the score observations obtained for each dataset, we obtain a posterior distribution which gives us the probabilities that one algorithm outperforms the other. We also introduce a *rope* (region of practically equivalent) region, in which we consider the algorithms to have equivalent performance. We have designated the rope region to be the one where the score differences are in the interval $[-0.01, 0.01]$. In summary, from the posterior distribution we obtain three probabilities: the probability that the first algorithm outperforms the second, the probability that the second algorithm outperforms the first one, and the probability that both algorithms are equivalent. These probabilities can be visualized in a simplex plot for a sample of the posterior distribution, in which a greater tendency of the points towards one of the regions will represent a greater probability.

To do the Bayesian sign tests, we have used the R package `rNPBST` [76]. In Figure 6 we pairwise compare some of the algorithms that seem to have better performance in experiment 1 with 3-NN (NCA, DMLMJ and LMNN) with the results of the 3-NN classifier for Euclidean distance. In the comparison made between Euclidean distance and NCA, we can clearly see that the points are concentrated close to the [NCA, rope] segment. This shows us that Euclidean distance is unlikely to outperform NCA, and there is also a high probability for NCA to outperform Euclidean distance, since a big concentration of points is in the NCA region. We obtain similar conclusions for DMLMJ against Euclidean distance, although in this case, despite the fact that Euclidean distance is still unlikely to win, there is a greater concentration of points in the rope region. In the comparison made between LMNN and Euclidean distance, we see a more centered concentration of points, that is slightly weighted towards the LMNN region. In the comparisons made between the DML algorithms we observe the points weighted to the [NCA, rope] segment, which concludes the difficulty of outperforming NCA, and between DMLMJ and LMNN we can see a pretty level playing field that is slightly biased to the DMLMJ algorithm.

The outperforming of Euclidean distance is even clearer in the results from experiment 2. For these algorithms, we can clearly observe that the points are concentrated in the region corresponding to the nearest centroid metric learning algorithm, as shown in Figure 7. We have elaborated more pairwise Bayesian sign tests for the rest of the algorithms in experiment 1. The results of these tests can also be found on the pyDML-Stats website[4].

5 Prospects and Challenges in Distance Metric Learning

Throughout this tutorial we have seen what DML consists of and how it has traditionally been applied in machine learning. However, the development of technology in recent years has given rise to new problems that cannot be adequately addressed from the point of view of classic machine learning. In the same way, this technological development has led to the creation new tools that are very useful when facing new problems, as well as allowing better results to be obtained with the more traditional problems.

Focusing on DML, both the new problems and the new tools are generating new prospects where in which applying DML could be of interest, and as well as generating new challenges in the design and application of DML. Below we will describe some of the most outstanding ones.

5.1 Prospects of Distance Metric Learning in Machine Learning

Nowadays there are many fields where the further development of DML might be of interest. On the one hand, the large volumes of data that are usually being handled today make it necessary to adapt or design new algorithms that can work properly with both high-dimensional data and huge amounts of examples. Similarly, new problems are arising, which make it necessary to reconsider the algorithms so that they can handle these problems in an appropriate way. On the other hand, many of the tools provided by machine learning, from the classical ones to the most modern ones, can be used in line with DML to achieve better results. We outline these prospects below.

- **Hybridization with feature selection techniques to solve high dimensional data problems.** DML is of great interest in many real problems in high dimensionality, such as face recognition, where it is very useful to be able to measure the similarity between different images [30]. When we work with datasets of even greater dimensionality, the treatment of distances can become too expensive, since it would be necessary to store matrices of very large dimensions. In these situations, it may be of interest to combine DML with feature selection techniques prepared for very high dimensional data [77, 26].

- **Big Data solutions.** The problem of learning when the amount of data we have is huge and heterogeneous is one of the challenges of machine learning nowadays [78]. In the case of the DML algorithms, although many of them, especially those based on gradient descent, are quite slow and do not scale well with the number of

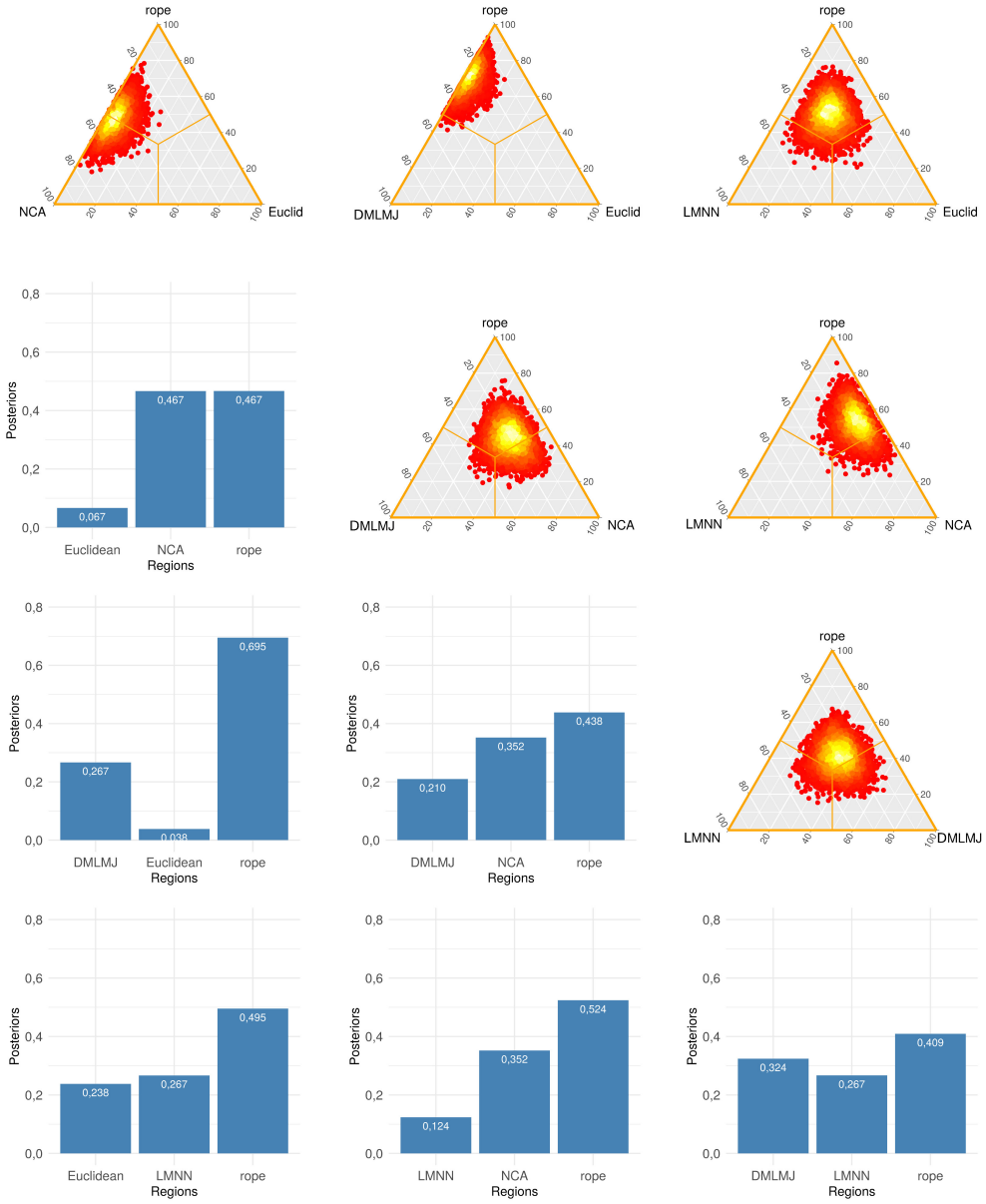


Figure 6: Bayesian sign results for NCA, DMLMJ, LMNN and Euclidean distance with 3-NN.

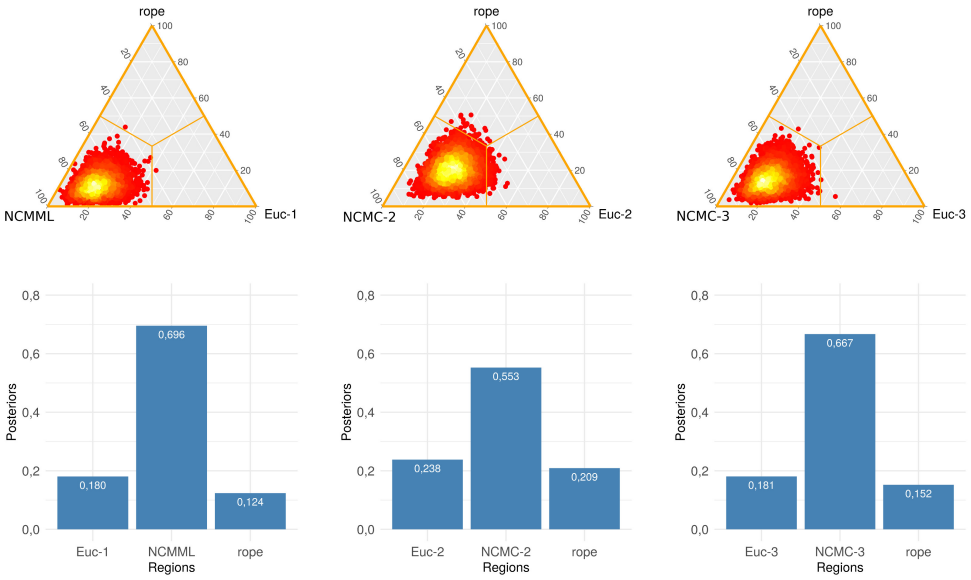


Figure 7: Bayesian sign test results for the comparison between scores of nearest centroid classifiers with their corresponding DML algorithm against the same classifier with Euclidean distance. The results are shown for nearest class mean classifier (left), nearest class with 2 centroids (center) and nearest class with 3 centroids (right).

samples, they can be largely parallelized in both matrix computations and gradient descent batches. As a result, DML can be extended to handle Big Data by developing specialized algorithms and integrating them with frameworks such as Spark [79] and Cloud Computing architectures [80].

- **Application of distance metric learning to singular supervised learning problems.** In this paper, we have focused on DML for common problems, like standard classification and dimensionality reduction, and we have also mentioned its applications for clustering and semi-supervised learning. However, DML can be useful in a wide variety of *non-standard* learning tasks [81], and can be carried out either by designing new algorithms or by adapting known algorithms from standard problems to these tasks. In recent years, several DML proposals have been made in problems like regression [82], multi-dimensional classification [83], ordinal classification [84], multi-output learning [85] and even transfer learning [86, 13].

- **Hybridization with shallow learning techniques.** Over the years, some distance-based algorithms, or some of their ideas or foundations, have been combined with other algorithms in order to improve their learning capabilities in certain problems. For example, the concept of nearest-neighbors has been combined with classifiers such as Naive-Bayes, obtaining a Naive-Bayes classifier whose feature distributions are determined by the nearest neighbors of each class [87]; with neural

networks, to find the best neural network architecture [88]; with random forests, by exploiting the relationship between voting points and potential nearest neighbors [89]; with ensemble methods, like bootstrap [90, 91]; with support vector machines, training them locally in neighborhoods [92]; or with rule-learning algorithms, obtaining the so-called *nested generalized exemplar* algorithms [93]. The distances used in these combinations of algorithms can condition their performance, so designing appropriate distance learning algorithms for each of these tasks can help achieve good results. Staying on this subject, another option is to hybridize directly DML with other techniques, like ensemble learning [94].

- **Hybridization with deep learning techniques.** In recent years, machine learning has experienced great popularity thanks to the development of deep learning, which is capable of obtaining very good results in different learning problems [95]. As in the previous case, it is possible to combine distance-based algorithms or their foundations with deep learning techniques to improve their learning capabilities. For instance, [96] use the k-nearest neighbors classifier to provide interpretability and robustness to deep neural networks. Another prospect that has gained popularity in recent years is based on the use of neural networks to learn distances, which is being referred to as *deep metric learning* [97, 98, 99, 100, 101]. Deep learning is likely to play an important role in the future of machine learning, and thus its combination with DML may lead to interesting advances in both fields.

- **Other approaches for the concept of distance.** Most of the current DML theory focus on Mahalanobis distances. However, some articles open a door to learning about other possible distances, such as local Mahalanobis distances, that lead to a multi-metric learning [38], or approaches beyond the Mahalanobis theory [102, 103]. The *deep metric learning* approach discussed above is another way of handling a wider range of distances. By developing new approaches, we will have a greater variety of distances to learn, and thus have a greater chance of success.

5.2 Challenges in Distance Metric Learning

In addition to the numerous action horizons, DML presents several challenges in terms of the design of its algorithms, which can lead to substantial improvements. We describe these challenges below.

- **Non-linear distance metric learning.** As we have already mentioned, since learning a Mahalanobis distance is equivalent to learning a linear map, there are many problems where these distances are not able to capture the inherent non-linearity of the data. Although the non-linearity of a subsequent learning algorithm, such as the nearest neighbors classifier, may mitigate this fact, that algorithm could benefit much more from a distance capable of capturing the non-linearity of the dataset. In this sense, we have already seen how the kernelization of DML algorithms can be applied to fit non-linear data. Extending the kernel trick to other

algorithms besides those presented, by searching for suitable parameterizations and representer theorems, is another possible task to carry out. Another possibility for non-linear DML is to adapt classical objective functions so that they can work with non-linear distances, such as the χ^2 histogram distance, or with non-linear transformations of the data learned by another algorithm, such as gradient boosting [104].

- **Multi-linear distance metric learning.** In learning problems where the data are images or videos, the traditional vector representation may not be the most appropriate to fit the data properly. Vector representation does not allow, for example, for the consideration of neighborhood relationships between pixels in an image. It is therefore better to consider images as matrices, or more generally, as multi-linear mappings or *tensors*. Some DML algorithms can be adapted so that they can learn distances in tensor spaces [105, 36, 106], which will be more suitable for similarity learning in datasets that support this representation. The development or extension of techniques for multi-linear DML is a challenge that has many applications in a field such as computer vision, where DML has been shown to be quite useful [17, 19, 21, 68].

- **Other optimization mechanisms.** The algorithms we have studied optimize their objective functions by applying gradient descent methods. However, the possibilities in terms of optimization mechanisms are very broad, and choosing the most appropriate method can contribute to achieving better values in the objective functions. In addition, the consideration of different optimization methods may lead to the design of new objective functions that may be appropriate for new problems or approaches and that cannot be optimized by classical gradient methods. In this way, we have studied several differentiable objective functions in this tutorial, unconstrained or with convex constraints, but for those non-convex functions the gradient descent methods (even the stochastic version) cannot guarantee a convergence to the global optimum. If we wanted to consider functions with even worse analytical characteristics or constraints, such as non-differentiability or integer constraints, we could not even use this type of method. For the non-convex and differentiable case, we are still able to use the information of the derivatives of the objective function, and some refinements of the classic gradient methods, such as AdaDelta, RMSprop or Adam have shown good performance in this type of problem [107]. In the most general case, we are only able to afford to evaluate the objective function, and sometimes not a very high number of times, due to its complexity. This general case is usually called *black-box optimization*. To optimize these functions, a wide variety of proposals have been made. If we cannot afford to evaluate the objective function many times, Bayesian optimization may be an interesting alternative [108]. If the objective function is not so complex, evolutionary algorithms can provide us with a great capability of exploration in the search space. Their repertoire is much broader and includes techniques such as *simulated annealing*, *particle swarm optimization* or *response surface methods*, among others [109], thus many tools are available to address the most diverse optimization problems. These heuristics can also be used

over differentiable optimization problems, and sometimes they can even outperform gradient methods, thanks to their greater ability to escape from local optima [110]. The evolutionary approach has been explored recently in several DML problems [111, 112].

6 Conclusions

In this tutorial we have studied the concept of distance metric learning, showing what it is, what its applications are, how to design its algorithms, and the theoretical foundations of this discipline. We have also studied some of the most popular algorithms in this field and their theoretical foundations, and explained different resolution techniques.

In order to understand the theoretical foundations of distance metric learning and its algorithms, it was necessary to delve into three different mathematical theories: convex analysis, matrix analysis and information theory. Convex analysis has it possible to present many of the optimization problems studied in the algorithms, along with some methods for solving them. Matrix analysis provided many useful tools to help understand this discipline, from how to parameterize Mahalanobis distances, to the optimization with eigenvectors, going through the most basic algorithm of semidefinite programming. Finally, information theory has motivated several of the algorithms we have studied.

In addition, several experiments have been developed that have allowed for the evaluation of the performance of the algorithms analyzed in this study. The results of these experiments have allowed us to observe how algorithms such as LMNN, DMLMJ, and especially NCA can considerably improve the nearest neighbors classification, and how centroid-based distance learning algorithms also improve their corresponding classifiers. We have also seen the wide variety of possibilities offered by kernel-based algorithms, and the advantages that an appropriate reduction of the dimensionality of the datasets can offer.

Acknowledgements

Our work has been supported by the research project TIN2017-89517-P and by a research scholarship (FPU18/05989), given to the author Juan Luis Suárez by the Spanish Ministry of Science, Innovation and Universities.

Compliance with ethical standards

Declaration of competing interest

The authors declare that there is no conflict of interest.

A Glossary of terms

ANMM Average Neighborhood Margin Maximization

DML Distance Metric Learning

DML-eig Distance Metric Learning with eigenvalue optimization

DMLMJ Distance Metric Learning through the Maximization of the Jeffrey divergence

ITML Information Theoretic Metric Learning

KANMM Kernel Average Neighborhood Margin Maximization

KDA Kernel Discriminant Analysis

KDMLMJ Kernel Distance Metric Learning through the Maximization of the Jeffrey divergence

KLMNN Kernel Large Margin Nearest Neighbors

***k*-NN** *k*-Nearest Neighbors

LDA Linear Discriminant Analysis

LDML Logistic Discriminant Metric Learning

LMNN Large Margin Nearest Neighbors

LSI Learning with Side Information

MCML Maximally Collapsing Metric Learning

MMC Mahalanobis Metric for Clustering

NCA Neighborhood Components Analysis

NCM Nearest Class Mean

NCMC Nearest Class with Multiple Centroids

NCMML Nearest Class Mean Metric Learning

PCA Principal Components Analysis

SVM Support Vector Machines

References

[1] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[2] George S Sebestyen. *Decision-making processes in pattern recognition*. Macmillan, 1962.

[3] Nils J Nilsson. *Learning machines: foundations of trainable pattern-classifying systems*. McGraw-Hill, 1965.

[4] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 521–528, 2003.

[5] Zongqing Ma, Shuang Zhou, Xi Wu, Heye Zhang, Weijie Yan, Shanhui Sun, and Jiliu Zhou. Nasopharyngeal carcinoma segmentation based on enhanced convolutional neural networks using multi-modal metric learning. *Physics in Medicine & Biology*, 64(2):025005, 2019.

[6] Guohui Wei, Min Qiu, Kuixing Zhang, Ming Li, Dejian Wei, Yanjun Li, Peiyu Liu, Hui Cao, Mengmeng Xing, and Feng Yang. A multi-feature image retrieval scheme for pulmonary nodule diagnosis. *Medicine*, 99(4), 2020.

[7] Tie Li, Gang Kou, and Yi Peng. Improving malicious urls detection via feature engineering: Linear and nonlinear space transformation methods. *Information Systems*, page 101494, 2020.

[8] Yong Luo, Han Hu, Yonggang Wen, and Dacheng Tao. Transforming device fingerprinting for wireless security via online multitask metric learning. *IEEE Internet of Things Journal*, 7(1):208–219, 2020.

[9] Yufei Liu, Dechang Pi, and Lin Cui. Metric learning combining with boosting for user distance measure in multiple social networks. *IEEE Access*, 5:19342–19351, 2017.

[10] Yang Liu, Zhonglei Gu, Tobey H Ko, and Jiming Liu. Multi-modal media retrieval via distance metric learning for potential customer discovery. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 310–317. IEEE, 2019.

[11] Ruirui Li, Jyun-Yu Jiang, Jiahao Liu Li, Chu-Cheng Hsieh, and Wei Wang. Automatic speaker recognition with limited data. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 340–348, 2020.

[12] Zhongxin Bai, Xiao-Lei Zhang, and Jingdong Chen. Speaker verification by partial auc optimization with mahalanobis distance metric learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2020.

[13] Daniel Lopez-Sanchez, Angélica González Arrieta, and Juan M Corchado. Visual content-based web page categorization with deep transfer learning and metric learning. *Neurocomputing*, 338:418–431, 2019.

[14] Haifeng Hu, Kun Wang, Chenggang Lv, Jiansheng Wu, and Zhen Yang. Semi-supervised metric learning-based anchor graph hashing for large-scale image retrieval. *IEEE Transactions on Image Processing*, 28(2):739–754, 2019.

[15] Hao Wu, Qimin Zhou, Rencan Nie, and Jinde Cao. Effective metric learning with co-occurrence embedding for collaborative recommendations. *Neural Networks*, 124:308–318, 2020.

[16] Xiaotong Li and Yan Tang. A social recommendation based on metric learning and network embedding. In *2020 IEEE 5th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*, pages 55–60. IEEE, 2020.

[17] B. Nguyen and B. De Baets. Kernel distance metric learning using pairwise constraints for person re-identification. *IEEE Transactions on Image Processing*, 28(2):589–600, 2019.

[18] C. Zhao, X. Wang, W. Zuo, F. Shen, L. Shao, and D. Miao. Similarity learning with joint transfer constraints for person re-identification. *Pattern Recognition*, 97, 2020.

[19] J. Liang, Q. Hu, C. Dang, and W. Zuo. Weighted graph embedding-based metric learning for kinship verification. *IEEE Transactions on Image Processing*, 28(3):1149–1162, 2019.

[20] F. Dornaika, I. Arganda-Carreras, and O. Serradilla. Transfer learning and feature fusion for kinship verification. *Neural Computing and Applications*, 32(11):7139–7151, 2020.

[21] Duo Wang, Yu Cheng, Mo Yu, Xiaoxiao Guo, and Tao Zhang. A hybrid approach with optimization-based and metric-based meta-learner for few-shot learning. *Neurocomputing*, 349:202–211, 2019.

[22] Chen Wang, Guohua Peng, and Bernard De Baets. Deep feature fusion through adaptive discriminative metric learning for scene recognition. *Information Fusion*, 2020.

[23] Yidi Du, Chang Liu, and Bo Zhang. Detection of gh pituitary tumors based on mnf. In *2019 Chinese Control And Decision Conference (CCDC)*, pages 635–639. IEEE, 2019.

[24] Jonathan R Wells, Sunil Aryal, and Kai Ming Ting. Simple supervised dissimilarity measure: Bolstering iforest-induced similarity with class information without learning. *Knowledge and Information Systems*, pages 1–14, 2020.

[25] B. Nguyen, C. Morell, and B. De Baets. Scalable large-margin distance metric learning using stochastic gradient descent. *IEEE Transactions on Cybernetics*, 50(3):1072–1083, 2020.

[26] Kuan Liu and Aurélien Bellet. Escaping the curse of dimensionality in similarity learning: Efficient frank-wolfe algorithm and generalization bounds. *Neurocomputing*, 333:185–199, 2019.

[27] Liu Yang and Rong Jin. Distance metric learning: A comprehensive survey. *Michigan State Universiy*, 2(2):4, 2006.

[28] Brian Kulis et al. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2013.

[29] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013.

[30] Panagiotis Moutafis, Mengjun Leng, and Ioannis A Kakadiaris. An overview and empirical comparison of distance metric learning methods. *IEEE transactions on cybernetics*, 47(3):612–625, 2017.

[31] Ralph Tyrell Rockafellar. *Convex analysis*. Princeton university press, 2015.

[32] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[33] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 1990.

[34] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.

[35] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[36] Fei Wang and Changshui Zhang. Feature extraction by maximizing the average neighborhood margin. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[37] John P Cunningham and Zoubin Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *Journal of Machine Learning Research*, 16(1):2859–2900, 2015.

[38] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[39] Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Ruslan R Salakhutdinov. Neighbourhood components analysis. In *Advances in neural information processing systems*, pages 513–520, 2005.

[40] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2624–2637, 2013.

[41] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216. ACM, 2007.

[42] Bac Nguyen, Carlos Morell, and Bernard De Baets. Supervised distance metric learning through maximization of the jeffrey divergence. *Pattern Recognition*, 64:215–225, 2017.

[43] Amir Globerson and Sam T Roweis. Metric learning by collapsing classes. In *Advances in neural information processing systems*, pages 451–458, 2006.

[44] Lorenzo Torresani and Kuang-chih Lee. Large margin component analysis. In *Advances in neural information processing systems*, pages 1385–1392, 2007.

[45] Sebastian Mika, Gunnar Ratsch, Jason Weston, Bernhard Scholkopf, and Klaus-Robert Mullers. Fisher discriminant analysis with kernels. In *Neural networks for signal processing IX, 1999. Proceedings of the 1999 IEEE signal processing society workshop.*, pages 41–48. IEEE, 1999.

[46] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *Journal of Machine Learning Research*, 18(1):2653–2688, 2017.

[47] Juan Luis Suárez, Salvador García, and Francisco Herrera. pydml: A python library for distance metric learning. *Journal of Machine Learning Research*, 21(96):1–7, 2020.

[48] Juan Luis Suárez, Salvador García, and Francisco Herrera. A tutorial on distance metric learning: Mathematical foundations, algorithms, experiments, prospects and challenges (with appendices on mathematical background and detailed algorithms explanation). *arXiv preprint arXiv:1812.05944*, 2020.

[49] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[50] Yingtong Dou, Hao Yang, and Xiaolong Deng. A survey of collaborative filtering algorithms for social recommender systems. In *2016 12th International Conference on Semantics, Knowledge and Grids (SKG)*, pages 40–46. IEEE, 2016.

[51] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie Mellon University, 2002.

[52] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

[53] Amir Ahmad and Lipika Dey. A method to compute distance between two categorical values of same attribute in unsupervised learning for categorical data set. *Pattern Recognition Letters*, 28(1):110–118, 2007.

[54] David B Blumenthal and Johann Gamper. On the exact computation of the graph edit distance. *Pattern Recognition Letters*, 134:46–57, 2020.

[55] Mohammad Norouzi, David J Fleet, and Russ R Salakhutdinov. Hamming distance metric learning. In *Advances in neural information processing systems*, pages 1061–1069, 2012.

[56] Lei Ma, Hongliang Li, Fanman Meng, Qingbo Wu, and King Ngi Ngan. Discriminative deep metric learning for asymmetric discrete hashing. *Neurocomputing*, 380:115–124, 2020.

[57] Alice Zheng and Amanda Casari. *Feature engineering for machine learning: principles and techniques for data scientists*. O'Reilly Media, Inc., 2018.

[58] Özgür Yeniay. Penalty function methods for constrained optimization with genetic algorithms. *Mathematical and computational Applications*, 10(1):45–56, 2005.

[59] Tianbao Yang, Qihang Lin, and Lijun Zhang. A richer theory of convex constrained optimization with reduced projections and improved rates. In *International Conference on Machine Learning*, pages 3901–3910, 2017.

[60] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[61] Effrosini Kokiopoulou, Jie Chen, and Yousef Saad. Trace optimization and eigenproblems in dimension reduction methods. *Numerical Linear Algebra with Applications*, 18(3):565–602, 2011.

[62] Charu C Aggarwal, Yuchen Zhao, and S Yu Philip. On text clustering with side information. In *Proceedings - International Conference on Data Engineering*, pages 894–904. IEEE, 2012.

[63] Paul S Bradley and Usama M Fayyad. Refining initial points for k-means clustering. In *International Conference on Machine Learning*, volume 98, pages 91–99. Citeseer, 1998.

[64] Philipp Probst, Anne-Laure Boulesteix, and Bernd Bischl. Tunability: Importance of hyperparameters of machine learning algorithms. *Journal of Machine Learning Research*, 20(53):1–32, 2019.

[65] Paramveer S Dhillon, Partha Pratim Talukdar, and Koby Crammer. Inference-driven metric learning for graph construction. In *4th North East Student Colloquium on Artificial Intelligence*, 2010.

[66] I.T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, 2002.

[67] Yiming Ying and Peng Li. Distance metric learning with eigenvalue optimization. *Journal of Machine Learning Research*, 13(Jan):1–26, 2012.

[68] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Is that you? metric learning approaches for face identification. In *2009 IEEE 12th International Conference on Computer Vision*, pages 498–505. IEEE, 2009.

[69] Michael L Overton. On minimizing the maximum eigenvalue of a symmetric matrix. *SIAM Journal on Matrix Analysis and Applications*, 9(2):256–268, 1988.

[70] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[71] Ratthachat Chatpatanasiri, Teesid Korsrilabutr, Pasakorn Tangchanachaianan, and Boonserm Kijsirikul. A new kernelization framework for mahalanobis distance learning algorithms. *Neurocomputing*, 73(10-12):1570–1579, 2010.

[72] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[73] Isaac Triguero, Sergio González, Jose M Moyano, Salvador García, Jesús Alcalá-Fdez, Julián Luengo, Alberto Fernández, Maria José del Jesús, Luciano Sánchez, and Francisco Herrera. Keel 3.0: an open source software for multi-stage analysis in data mining. *International Journal of Computational Intelligence Systems*, 10(1):1238–1249, 2017.

[74] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[75] Alessio Benavoli, Giorgio Corani, Francesca Mangili, Marco Zaffalon, and Fabrizio Ruggeri. A bayesian wilcoxon signed-rank test based on the dirichlet process. In *International conference on machine learning*, pages 1026–1034, 2014.

[76] Jacinto Carrasco, Salvador García, María del Mar Rueda, and Francisco Herrera. rnpbst: An r package covering non-parametric and bayesian statistical tests. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 281–292. Springer, 2017.

[77] Mingkui Tan, Ivor W Tsang, and Li Wang. Towards ultrahigh dimensional feature selection for big data. *Journal of Machine Learning Research*, 15(1):1371–1429, 2014.

[78] Xindong Wu, Xingquan Zhu, Gong-Qing Wu, and Wei Ding. Data mining with big data. *IEEE transactions on knowledge and data engineering*, 26(1):97–107, 2014.

[79] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, et al. Mllib: Machine learning in apache spark. *Journal of Machine Learning Research*, 17(1):1235–1241, 2016.

[80] Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan. The rise of "big data" on cloud computing: Review and open research issues. *Information systems*, 47:98–115, 2015.

[81] David Charte, Francisco Charte, Salvador García, and Francisco Herrera. A snapshot on nonstandard supervised learning problems: taxonomy, relationships, problem transformations and algorithm adaptations. *Progress in Artificial Intelligence*, 8(1):1–14, 2019.

[82] Bac Nguyen, Carlos Morell, and Bernard De Baets. Large-scale distance metric learning for k-nearest neighbors regression. *Neurocomputing*, 214:805–814, 2016.

[83] Zhongchen Ma and Songcan Chen. Multi-dimensional classification via a metric approach. *Neurocomputing*, 275:1121–1131, 2018.

[84] Bac Nguyen, Carlos Morell, and Bernard De Baets. Distance metric learning for ordinal classification based on triplet constraints. *Knowledge-Based Systems*, 142:17–28, 2018.

[85] W. Liu, D. Xu, I.W. Tsang, and W. Zhang. Metric learning for multi-output tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):408–422, 2019.

[86] Y. Luo, Y. Wen, T. Liu, and D. Tao. Transferring knowledge fragments for learning distance metric from a heterogeneous domain. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(4):1013–1026, 2019.

[87] Xiaodong Yang and Ying Li Tian. Eigenjoints-based action recognition using naive-bayes-nearest-neighbor. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 14–19. IEEE, 2012.

[88] Lin Wang, Bo Yang, Yuehui Chen, Xiaoqian Zhang, and Jeff Orchard. Improving neural-network classifiers using nearest neighbor partitioning. *IEEE transactions on neural networks and learning systems*, 28(10):2255–2267, 2017.

[89] Yi Lin and Yongho Jeon. Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, 101(474):578–590, 2006.

[90] Brian M Steele. Exact bootstrap k-nearest neighbor learners. *Machine Learning*, 74(3):235–255, 2009.

[91] Yoshihiko Hamamoto, Shunji Uchimura, and Shingo Tomita. A bootstrap technique for nearest neighbor classifier design. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):73–79, 1997.

[92] Hao Zhang, Alexander C Berg, Michael Maire, and Jitendra Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2126–2136. IEEE, 2006.

[93] Dietrich Wettschereck and Thomas G Dietterich. An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. *Machine learning*, 19(1):5–27, 1995.

[94] Yang Mu, Wei Ding, and Dacheng Tao. Local discriminative distance metrics ensemble learning. *Pattern Recognition*, 46(8):2337–2349, 2013.

[95] Anabel Gómez-Ríos, Siham Tabik, Julián Luengo, ASM Shihavuddin, Bartosz Krawczyk, and Francisco Herrera. Towards highly accurate coral texture images classification using deep convolutional neural networks and data augmentation. *Expert Systems with Applications*, 118:315–328, 2019.

[96] Nicolas Papernot and Patrick McDaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*, 2018.

[97] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Deep metric learning for person re-identification. In *2014 22nd International Conference on Pattern Recognition*, pages 34–39. IEEE, 2014.

[98] Xuefei Zhe, Shifeng Chen, and Hong Yan. Directional statistics-based deep metric learning for image classification and retrieval. *Pattern Recognition*, 93:113–123, 2019.

[99] Fatih Cakir, Kun He, Xide Xia, Brian Kulis, and Stan Sclaroff. Deep metric learning to rank. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1861–1870, 2019.

[100] Xianghai Cao, Yiming Ge, Renjie Li, Jing Zhao, and Licheng Jiao. Hyperspectral imagery classification with deep metric learning. *Neurocomputing*, 356:217–227, 2019.

[101] Bac Nguyen and Bernard De Baets. Improved deep embedding learning based on stochastic symmetric triplet loss and local sampling. *Neurocomputing*, 402:209–219, 2020.

[102] Jiajun Pan and Hoel Le Capitaine. Metric learning with submodular functions. *Neurocomputing*, 2020.

[103] Hikaru Shindo, Masaaki Nishino, Yasuaki Kobayashi, and Akihiro Yamamoto. Metric learning for ordered labeled trees with pq -grams. In *24th European Conference of Artificial Intelligence*, 2020.

[104] Dor Kedem, Stephen Tyree, Fei Sha, Gert R Lanckriet, and Kilian Q Weinberger. Nonlinear metric learning. In *Advances in neural information processing systems*, pages 2573–2581, 2012.

[105] Deng Cai, Xiaofei He, Jiawei Han, Deng Cai, Xiaofei He, and Jiawei Han. Subspace learning based on tensor analysis. Technical report, 2005.

[106] Oualid Laiadi, Abdelmalik Ouamane, Abdelhamid Benakcha, Abdelmalik Taleb-Ahmed, and Abdenour Hadid. Tensor cross-view quadratic discriminant analysis for kinship verification in the wild. *Neurocomputing*, 377:286–300, 2020.

[107] Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao. A survey of optimization methods from a machine learning perspective. *IEEE transactions on cybernetics*, 2019.

[108] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

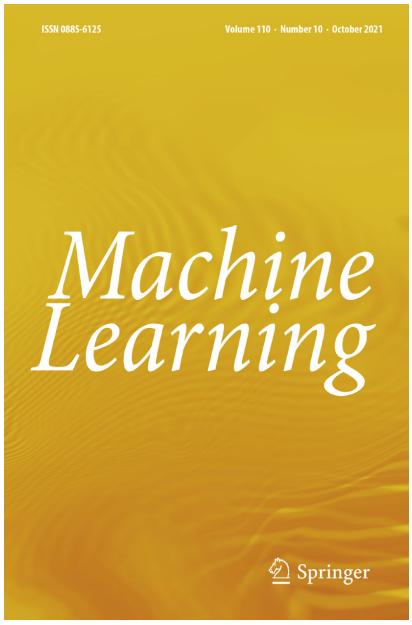
[109] Luis Miguel Rios and Nikolaos V Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, 2013.

[110] Gregory Morse and Kenneth O Stanley. Simple evolutionary optimization can rival stochastic gradient descent in neural networks. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 477–484, 2016.

[111] Wasin Kalintha, Satoshi Ono, Masayuki Numao, and Ken-ichi Fukui. Kernelized evolutionary distance metric learning for semi-supervised clustering. In *31st AAAI Conference on Artificial Intelligence*, pages 4945–4946, 2017.

[112] Bassel Ali, Wasin Kalintha, Koichi Moriyama, Masayuki Numao, and Ken-ichi Fukui. Reinforcement learning for evolutionary distance metric learning systems improvement. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 155–156. ACM, 2018.

3 Ordinal regression with explainable distance metric learning based on ordered sequences



- **Journal:** Machine Learning (MACH)
- **JCR Impact Factor:** 5.414
- **Rank:** 40/145
- **Quartile:** Q2
- **Category:** Computer Science, Artificial Intelligence
- **Status:** Published

Ref.: Suárez, J. L., García, S. & Herrera, F. (2021). Ordinal regression with explainable distance metric learning based on ordered sequences. Machine Learning, 110(10), 2729-2762. DOI: https://doi.org/10.1007/s10994-021-06010-w.

An extended abstract of this paper was published at the 2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA) with title: Ordinal Regression with Explainable Distance Metric Learning Based on Ordered Sequences: Extended Abstract (DOI: https://doi.org/10.1109/DSAA53316.2021.9564133). This conference was held thematically due to the COVID-19 pandemic.

ORDINAL REGRESSION WITH EXPLAINABLE DISTANCE METRIC LEARNING BASED ON ORDERED SEQUENCES

Juan Luis Suárez [*,a,b] **Salvador García** [a,b] **Francisco Herrera** [a,b]

[a] *Department of Computer Science and Artificial Intelligence (DECSAI), University of Granada, Granada, Spain*
[b] *Andalusian Institute of Data Science and Computational Intelligence (DaSCI)*

ABSTRACT

The purpose of this paper is to introduce a new distance metric learning algorithm for ordinal regression. Ordinal regression addresses the problem of predicting classes for which there is a natural ordering, but the real distances between classes are unknown. Since ordinal regression walks a fine line between standard regression and classification, it is a common pitfall to either apply a regression-like numerical treatment of variables or underrate the ordinal information applying nominal classification techniques. On a different note, distance metric learning is a discipline that has proven to be very useful when improving distance-based algorithms such as the nearest neighbors classifier. In addition, an appropriate distance can enhance the explainability of this model. In our study we propose an ordinal approach to learning a distance, called *chain maximizing ordinal metric learning*. It is based on the maximization of ordered sequences in local neighborhoods of the data. This approach takes into account all the ordinal information in the data without making use of any of the two extremes of classification or regression, and it is able to adapt to data for which the class separations are not clear. We also show how to extend the algorithm to learn in a non-linear setup using kernel functions. We have tested our algorithm on several ordinal regression problems, showing a high performance under the usual evaluation metrics in this domain. Results are verified through Bayesian non-parametric testing. Finally, we explore the capabilities of our algorithm in terms of explainability using the case-based reasoning approach. We show these capabilities empirically on two different datasets, experiencing signif-

* Corresponding Author (jlsuarezdiaz@decsai.ugr.es)
Email addresses: salvagl@decsai.ugr.es (Salvador García), herrera@decsai.ugr.es (Francisco Herrera)

icant improvements over the case-based reasoning with the traditional Euclidean nearest neighbors.

Keywords Distance Metric Learning · Ordinal regression · Nearest neighbors.

1 Introduction

Ordinal regression is a machine learning task which aims to predict variables that take values in an ordered and non-numerical set. It is an interesting research topic with applications in many areas, such as weather forecasting [1], medicine [2, 3, 4], psychology [5], social media mining [6] or computer vision [7, 8, 9]. This type of problem arises in many situations with human labelers, since they usually prefer to quantify data using categorical values rather than continuous values. The key element that differentiates ordinal regression from standard classification and regression is the presence of an order relation among the labels and of unquantifiable differences between two consecutive labels. This makes it necessary to adapt the traditional classification and regression techniques or to design new approaches to deal with ordinal regression problems in an appropriate way [10].

Since ordinal regression walks a fine line between classification and regression, it was traditionally common to approach the problem from one of these naive perspectives. This could lead to suboptimal approaches, as they might not take adequate advantage of the ordinal information in the data. Over time, the problem of ordinal regression has become increasingly important, due to its presence in many real problems, and new methods have been developed. [11] propose to decompose the ordinal regression problem into binary subproblems considering the inequalities with each of the possible classes. For each subproblem a binary classifier is trained and finally the prediction probabilities are aggregated to obtain the final ordinal class. [12] propose a generalization of the perceptron for ordinal regression, modifying the output layer to handle ordinal labels. [13] extend support vector machines for ordinal regression problems by imposing constraints on the classes, both explicit and implicit. [14] propose an original method to address this problem, which consists in adding additional dimensions to the data and replicating them in these dimensions. In this new scenario, the problem is transformed into a binary problem where the binary label assigned to each dimension will depend on the original ordinal class. The problem then can be solved using a binary classifier. [15] extend the AdaBoost ensemble to adapt it to ordinal classes. More recently, [16] propose an encoding and a kernel version of extreme learning machines to deal with ordinal regression problems, and [17] extend the Bayesian network classifiers by maximizing several metrics that better adapt to imbalanced and ordinal problems.

Learning using similarities between data is quite effective and adaptable to a wide variety of problems. It is inspired by the human ability to detect similarities among different objects, and it is one of the oldest practices carried out in machine learning [18]. Since similarity-based learning is based on human reasoning, it also allows us to develop explainable models for which their learned knowledge can be interpreted [19, 20]. Similarity-based approaches need to establish a similarity measure, or equivalently, a distance metric, among the data. Typically, standard distances, such as the Euclidean distance, are used for this purpose.

However, standard distances may not fit our data as well as a distance that has been learned from the dataset itself. Learning a suitable distance that allows us to facilitate a subsequent knowledge extraction is the task of distance metric learning [21]. Distance metric learning has proven to be highly effective for many different problems in machine learning, such as multi-dimensional classification [22] and multi-output learning [23].

Distance metric learning can be really useful for ordinal regression, since the learned distances can help to capture the ordinal information of the data. By learning a distance, we can get similar data to have similar labels as well, according to the order relation among the classes. In this way, the subsequent similarity learning will be more effective.

Several proposals to learn distances for ordinal regression problems have been formulated [24, 25], but they use the differences between classes internally, which may make them more appropriate for standard regression problems.

A more recent proposal [26] attempts to overcome this drawback by drawing inspiration from one of the most popular distance metric learning algorithms, LMNN [27]. This algorithm relies on the large margin principle to locally bring same-class data closer together, while preventing data from other classes from getting close to a specific limit. The extension to the ordinal case is made by adding further constraints so that, for a given sample, the samples with the most distant classes in the output space cannot be brought as close together in the input space as others with closer labels. This algorithm also supports a kernel version that is able to learn a distance in very high dimensional spaces. We will refer to the linear and kernel version of this proposal as LODML and KODML, respectively. However, seeking an airtight separation of classes, like this algorithm does, may be undesirable in many ordinal regression problems. Ordinal datasets are usually small and suffer from a high degree of subjectivity in their labeling [28]. Examples of this subjectivity are datasets that are generated from *likert scales* [29] or pain intensity scales for medical purposes. In these cases the perception of the classes may be different for each labeler. This results in heterogeneous datasets with unclear class boundaries. The underlying order is what actually contains the information in the dataset, rather than the true classes assigned to each sample.

Therefore, two of the main drawbacks from which the existing distance metric learning algorithms tend to suffer when applied to ordinal regression are:

- The quantification of differences between classes during the operation of the algorithms.
- The assumption that classes in ordinal datasets can be easily separated.

In this paper we propose a new distance metric learning algorithm for ordinal regression that overcomes these drawbacks. Our algorithm, which we have called *chain maximizing ordinal metric learning* (CMOML), is based on the premise that no matter how heterogeneous our dataset is, a distance-based classifier can better predict a sample when labels gradually become more and more different as we move away from the sample in any direction. In order to support our proposal, Figure 1 illustrates a good data layout for distance-based

ordinal regression. This figure is a snapshot of a bidimensional dataset around the point $x \in \mathbb{R}^2$. We can see here that, although there is not a clear boundary between each class in the dataset, when we move away from x in most directions we notice that the labels begin to increase (marked with the red dashed arrows) or decrease (marked with the blue dashed arrows) gradually. Therefore, a distance-based classifier is expected to perform well around x. CMOML is based on this idea in order to transform the data such that a good arrangement for a subsequent distance-based learning can be achieved. If many same-class samples are found, they will move closer to one another. If there is more variety, an optimal local sorting will be sought like in Figure 1.

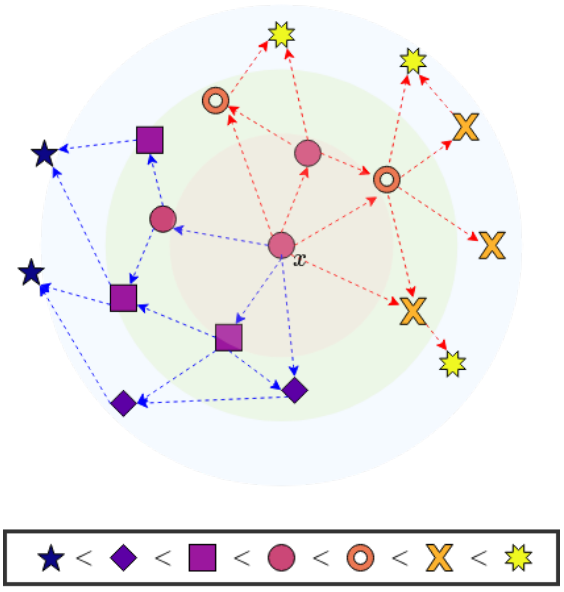


Figure 1: A snapshot of a 2D ordinal dataset around the point x. When we move away from x in most directions, the labels gradually increase or decrease. This benefits similarity-based or distance-based classification around x.

CMOML looks for a distance that respects the local ordinal trend in our dataset as much as possible. This is achieved by introducing the concept of ordered sequences. For each sample we take a neighborhood from which we construct two types of sequences around the sample: sequences of samples, in the input space, and the corresponding sequences of labels, in the output space. An order relation can be established in both types of sequences, and when a pair of sequences is ordered in both the input and output space, we will refer to that pair as a chain. The goal of CMOML is to maximize the number of chains in the dataset. The configuration of the algorithm also allows it to be applied in high dimensional spaces with the support of the kernel functions. We will refer to the kernel version of CMOML as KCMOML.

The design of CMOML makes it possible to solve the problems of the previous proposals, since a) the labels are not treated numerically, as the sequences will only consider the rel-

ative positions between them, and b) the sequential approach fits the structure of the data regardless of the quality of the class separation.

In order to analyze the time complexity of CMOML we have performed a computational analysis of the algorithm, which includes a comparison with the time complexity of the state-of-the-art of metric learning for ordinal regression. The analysis shows that the compared algorithms have similar time complexities and, due to the nature of CMOML, it can highly benefit from parallelization.

To evaluate the performance of CMOML we have carried out several experiments on datasets for ordinal regression. The results, supported by a Bayesian statistical analysis, show that CMOML is an outstanding algorithm within the family of distance metric learning algorithms for ordinal regression, in addition to being competitive with the ordinal regression state-of-the-art.

CMOML also offers benefits in terms of explainability that the compared algorithms lack. The case-based reasoning approach [30] allows nearest neighbors-based algorithms to show their learned knowledge in an understandable way [20, 19]. By combining nearest neighbors with the distance that CMOML learns, we can make the neighbors we obtain much more intuitive to our human reasoning. We will show this empirically. To do this, we will analyze on two different datasets how the nearest neighbors behave with the Euclidean distance and with the distance learned by CMOML. Additionally we will see that CMOML can perform dimensionality reduction and we will explore its benefits in terms of understandability.

Our paper is organized as follows. Section 2 describes the distance metric learning and ordinal regression problems. Section 3 shows our proposal of distance metric learning for ordinal regression. Section 4 describes the experiments developed to evaluate the performance of our algorithm, and the results obtained. Section 5 shows how to combine CMOML and case-based reasoning to obtain explainable models and discusses the benefits of this approach on two different datasets. Finally, Section 6 ends with the concluding remarks.

2 Background

In this section we will describe the problems and tools that will be used throughout the paper.

2.1 Distance metric learning

Distance metric learning [21] is a discipline of machine learning that aims to learn distances from the data, where distance refers to a map $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, with $\mathcal{X}$ a non-empty set, satisfying the following conditions:

1. Coincidence: $d(x, y) = 0 \iff x = y$, for every $x, y \in \mathcal{X}$.
2. Symmetry: $d(x, y) = d(y, x)$, for every $x, y \in \mathcal{X}$.
3. Triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$, for every $x, y, z \in \mathcal{X}$.

Distance metric learning frequently focuses on learning Mahalanobis distances, since they are parameterized by matrices, and therefore they are easy to handle from a computational perspective. Given a positive semidefinite (also called metric) matrix M of dimension d, the Mahalanobis distance associated with M, for each $x, y \in \mathbb{R}^d$, is given by

$$d_M(x, y) = \sqrt{(x - y)^T M (x - y)}.$$

Since every metric matrix M can be decomposed as $M = L^T L$, where L is a matrix with the same number of columns as M, and verifying that $d_M(x, y) = \|L(x - y)\|_2$, a Mahalanobis distance can also be understood as the Euclidean distance after applying the linear map defined by the matrix L. Thus, distance metric learning comes down to learning a metric matrix M or to learning a linear map matrix L. Both approaches are equally valid, and each one has different advantages. For instance, learning M usually leads to convex optimization problems, while learning L can be used to guarantee a dimensionality reduction with no additional cost.

2.2 Ordinal regression and similarity approaches

Ordinal regression [10] is a machine learning problem consisting in, similarly to classification and regression, predicting the output label $y \in \mathcal{Y}$ of an input vector $x \in \mathbb{R}^d$, given a training set of labeled samples $S = \{(x_1, y_1), \dots, (x_N, y_N)\}$, with $x_1, \dots, x_N \in \mathbb{R}^d$ and $y_1, \dots, y_N \in \mathcal{Y}$. What differentiates ordinal regression from classification and regression is the nature of the set $\mathcal{Y}$. $\mathcal{Y}$ is a finite set $\mathcal{Y} = \{l_1, \dots, l_C\}$ of length $C \geq 3$, over which an order relation $<$ is defined so that $l_i < l_{i+1}$, for $i = 1, \dots, C - 1$. In addition, although the values in $\mathcal{Y}$ are ordered, their differences are not quantifiable, that is, the operations $l_i - l_j$ are not defined, for any $i, j \in 1, \dots, C$. Therefore, the exclusive information that $\mathcal{Y}$ provides in ordinal regression problems is the relative positions between the labels. It is often common to represent $\mathcal{Y}$ as $\mathcal{Y} = \{1, \dots, C\}$ together with the order relation of the natural numbers, always taking into account that the differences between the values in $\mathcal{Y}$ should not be used in this problem.

In ordinal regression, committing a prediction error with a class close to the real one in $\mathcal{Y}$ is not as serious as committing an error predicting the class furthest away from the true one. Therefore, both the algorithms and the evaluation metrics have to be adapted to face this problem and evaluate their solutions adequately. For this purpose, traditional methods such as support vector machines have been adapted to this problem by imposing constraints in order to respect the ordering of the classes [13, 31], or by reducing to binary problems in an appropriate way [32]. Other proposals try to adapt conventional loss or gain functions in order to handle ordinal data in the best possible way. These functions have been tested with algorithms such as decision trees [33], Bayesian networks [17] or extreme learning machines [16], as well as with gradient-based learners [34, 35], such as back-propagation neural networks. Finally, deep learning has also taken a leading role in this problem, with proposals that are mainly applied in fields related to computer vision [36, 4, 8].

In the context of similarity-based learning, the *k-nearest neighbors* (k-NN) approach [18] can be easily extended to the ordinal case. To do this, we have to consider the general aggregation function for a k-NN algorithm [37, 38], given by

$$f(x) = \arg\min_{y \in \mathcal{Y}} \sum_{j=1}^{k} w_j \, p(y_{i_j}, y),$$

where x is the sample to be predicted, i_j, $j = 1, \ldots, k$ are the indices of the k-nearest neighbors of x in the training set, w_j are weights to be considered for each neighbor, and p is a penalty function that measures how wrong it might be to label x with each of the classes, according to the information given for each of the neighbors. Observe that if all the weights are equal and $p(y, y^*) = [\![y \neq y^*]\!]$, f becomes the classic majority-vote aggregation function of the k-NN (here, $[\![\cdot]\!]$ denotes the indicator function for the condition inside the brackets). When the labels are ordinal, we can use additional penalty functions that take into account the order relation between the labels. According to [39], one of the most popular penalty functions is the L_1 penalty[1], given by $p(y, y^*) = |y - y^*|$. This penalty function leads to the *median-vote* or *weighted-median-vote* k-NN approaches, both of them being immediate extensions of this algorithm to ordinal regression.

In the context of distance metric learning for ordinal regression, [26] recently proposed the distance metric learning approach LODML (*linear ordinal distance metric learning*). This approach is based on the classic *large margin nearest neighbors* (LMNN) algorithm for distance metric learning. LMNN searches for a distance that brings each sample as close as possible to a predefined set of same-class *target neighbors* while keeping data from other classes out of a *large margin* defined by the target neighbors. This distance can be obtained by optimizing the following objective function, defined for a positive semidefinite matrix M:

$$\begin{aligned} f_L(M) &= \alpha \operatorname{tr}(M) + \sum_{(i,j,l) \in R_1} \xi_{ijl} \\ s.t. &: d_M^2(x_i, x_l) - d_M^2(x_i, x_j) \geq 1 - \xi_{ijl} \\ &\xi_{ijl} \geq 0 \\ &M \geq 0. \end{aligned}$$

In this function, the first term is a regularization term and ξ_{ijl} slack variables that measure to what degree the margins have been violated. The set R_1 is the set of triplets (i, j, l) such as x_j is a target neighbor of x_i and x_l is a sample with a different class to x_i. The first constraint is the *large margin constraint*, which enforces that, for each triplet $(i, j, l) \in R_1$, x_i and x_j are close, and x_l does not invade the large margin defined by x_j around x_i, as long as it is feasible.

[1] Observe that, although this penalty function is considering the differences between the labels, their numerical values do not influence the final value of the aggregation function. In fact, the median or weighted-median resulting from the aggregation does not depend on the differences between the labels, but depends only on the weights w_j and the relative positions between the labels of the k-nearest neighbors.

The extension to the ordinal case that LODML performs is done by replacing the set of triplets R_1 with a new set $R = R_1 \cup R_2 \cup R_3$, where R_1 is the previous set of triplets, and R_2 and R_3 include triplets (i, j, l) so that $y_l < y_j < y_i$ or $y_i < y_j < y_l$, respectively, thus forcing the large margin criterion to also be applied with all the possible combinations of ordered classes. The algorithm can be extended to non-linear metric learning using kernel functions. The kernel version is referred to as *kernel ordinal distance metric learning* (KODML).

3 Chain Maximizing Ordinal Metric Learning

In this section we will describe the CMOML algorithm. This algorithm looks for a distance for which the dataset has a high number of chains. As such, it achieves an optimal configuration for the data, since close samples according to the distance obtained will also have close labels.

First, we will explain in detail the concepts needed to understand the algorithm, Then, we will show the objective function and how to proceed with its calculation. Afterwards, we will demonstrate how to extend the algorithm to high dimensional spaces using kernels. We will conclude the section by performing a complexity analysis of the proposed methods.

3.1 Preliminary Definitions

Suppose that our training set is given by $\mathcal{X} = \{x_1, \dots, x_N\} \subset \mathbb{R}^d$, with corresponding labels $y_1, \dots, y_N \in \mathcal{Y} = \{1, \dots, C\}$. As already mentioned, given $x_i \in \mathcal{X}$, the goal of our algorithm is to increase the difference between the labels as we move away from x_i in any direction, for each $x_i \in \mathcal{X}$. To do this we introduce several concepts. First, we will work with the distance defined by a linear map $L : \mathbb{R}^d \to \mathbb{R}^{d'}$, with $d' \leq d$. The value of d' is the desired dimension of our dataset in the transformed space, so we are able to perform a dimensionality reduction with this setup. The associated matrix, which we will also refer to as L, is of dimension $d' \times d$. A common value for d' is to use $d' = d$, so that the transformed data remains in the input space. If the transformation learned in this way is not full-rank, that is, all the elements in $\mathbb{R}^d$ are mapped to a subspace of dimension lower than d, this dimension can be used as a new value for d' to ensure that no information from the full-rank method is lost. Another possibility is to estimate d' from specialized dimensionality reduction methods. For example, to set d' according to the reconstruction errors provided by *principal components analysis* [40], up to the amount of reconstruction error we are willing to admit for our data.

Then, for each $x_i \in \mathcal{X}$ we define a *neighborhood* $\mathcal{U}_L(x_i)$, which consists of the K_i nearest neighbors of x_i (including x_i), according to the distance determined by L. $K_i \in \mathbb{N}$ denotes the size of the neighborhood around x_i, and can be either a fixed value for every $i \in \{1, \dots, N\}$, a value estimated to make equal-radius neighborhoods that better adapts to the data densities, or any value that can be established with any kind of prior information. We will consider these neighbors as already projected onto the transformed space, that is, the elements of $\mathcal{U}_L(x_i)$ will be of the form Lx_j, with $j \in \{1, \dots, N\}$. We also fix $\kappa \in \mathbb{N}$, which will be the length of the sequences that we define below.

Definition. *Let σ be an injective mapping from the set $\{1, \dots, \kappa\}$ to the set of the indices of the samples that belong to $U_L(x_i)$. A* sequence (of samples) *around $x_i \in \mathcal{X}$ (with respect to L) is a κ-tuple $(Lx_{\sigma(1)}, \dots, Lx_{\sigma(k)})$, with $Lx_{\sigma(1)}, \dots, Lx_{\sigma(\kappa)} \in \mathcal{U}_L(x_i)$. We refer to the tuple $(y_{\sigma(1)}, \dots, y_{\sigma(\kappa)})$ as the corresponding* sequence of labels.

In other words, sequences of samples are successions of κ items taken from $U_L(x_i)$ without replacement, and their labels in the same order of choice form the corresponding sequence of labels. Since we are working on an ordinal regression problem, there is a natural order relation among the labels. We will say that a sequence of labels $(y_{\sigma(1)}, \dots, y_{\sigma(\kappa)})$ around x_i is *ordered* if either $y_i \leq y_{\sigma(1)} \leq \dots \leq y_{\sigma(\kappa)}$ or $y_i \geq y_{\sigma(1)} \geq \dots y_{\sigma(\kappa)}$.

The next step is to establish a relation on the sequences of samples so that they can be paired with the corresponding order in the sequences of labels. As we are interested in increasing the difference between the labels while we are moving away from x_i, we define this (pre-)order relation as follows.

Definition. *An* ordered sequence of samples *around $x_i \in \mathcal{X}$ (with respect to L) is a sequence of samples $(Lx_{\sigma(1)}, \dots, Lx_{\sigma(\kappa)})$ verifying that*

$$\|Lx_{\sigma(1)} - Lx_i\| \leq \|Lx_{\sigma(2)} - Lx_i\| \leq \dots \leq \|Lx_{\sigma(\kappa)} - Lx_i\|.$$

Definition. *A* chain *around $x_i \in \mathcal{X}$ (with respect to L) is an ordered sequence of labels $(y_{\sigma(1)}, \dots, y_{\sigma(\kappa)})$ associated with an ordered sequence of samples $(Lx_{\sigma(1)}, \dots, Lx_{\sigma(\kappa)})$. We denote the total number of chains around x_i (with respect to L) as $S_L(x_i)$.*

In short, chains represent sequences that are ordered in both the label space and the sample space. Observe that, if we assume that there are no points in $U_L(x_i)$ at the exactly same distance from x_i we have that

$$S_L(x_i) \leq \binom{K_i}{\kappa}, \tag{1}$$

since, as there is a unique ordering of the samples under this assumption, the total number of ordered sequences of samples is the total number of non-repeating combinations of κ items taken from $U_L(x_i)$, and $S_L(x_i)$ attains its maximum when every ordered sequence of samples is a chain. The case with equal distances will be discussed later on. Let us illustrate these concepts with an example.

Example. *Suppose that we set $K_0 = 7$, $\kappa = 3$, and, for a linear map $L : \mathbb{R}^d \to \mathbb{R}^2$, we have a snapshot of a 2D ordinal dataset around the point Lx_0 as in Figure 2. Under these assumptions we have that $U_L(x) = \{Lx_0, Lx_1, Lx_2, Lx_3, Lx_4, Lx_5, Lx_6\}$, and the distance orders in the neighborhood verify that $\|Lx_j - Lx\| < \|Lx_{j+1} - Lx\|$ for $j = 0, \dots, 6$. The labels corresponding to each of the samples in $U_L(x_0)$ are: $y_0 =$ ●, $y_1 =$ ◎, $y_2 =$ ✕, $y_3 =$ ■, $y_4 =$ ■, $y_5 =$ ◆ and $y_6 =$ ●. We have that:*

- *The sequences of samples are any κ different items taken from $U_L(x)$ in any order. For example, (Lx_5, Lx_0, Lx_3) (the gray dotted arrows). This sequence is not ordered, since Lx_0 and Lx_3 are closer to Lx_0 than the first element of the sequence, Lx_5. Its*

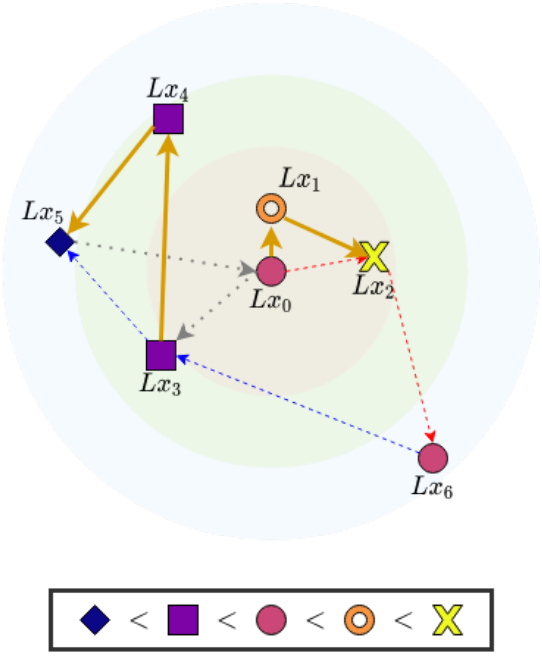


Figure 2: A neighborhood of the sample Lx_0, for L of length $K_0 = 7$, with sequences of samples and labels (gray dotted arrows), ordered sequences of samples (red dashed arrows), ordered sequences of labels (blue dashed arrows) and chains (golden solid arrows) of length $\kappa = 3$.

corresponding sequence of labels is $(y_5, y_0, y_3) =$ (◆, ●, ■)*, which is also not ordered, since neither* $y_5 \leq y_0 \leq y_3$ *nor* $y_5 \geq y_0 \geq y_3$.

- *The sequence of samples* (Lx_0, Lx_2, Lx_6) *(the red dashed arrows) is ordered, since each term in the sequence is further away from* Lx_0 *than the previous one. However, it is not a chain, since its corresponding sequence of samples is* $(y_0, y_2, y_6) =$ (●, X, ●)*, and this sequence is not ordered.*

- *The sequence of labels* $(y_6, y_3, y_5) =$ (●, ■, ◆) *(the blue dashed arrows) is ordered, since it is decreasing* (● $\geq$ ■ $\geq$ ◆) *and all the elements in the sequence are lower than or equal to the label of the central sample of the neighborhood,* $y_0 =$ ●*. However, it is not associated with an ordered sequence of samples, because in* (Lx_6, Lx_3, Lx_5)*, the second item is closer to* Lx_0 *than the first one. Therefore, this sequence is not a chain.*

- *The sequences of samples* (Lx_0, Lx_1, Lx_2) *and* (Lx_3, Lx_4, Lx_5) *are chains, since they are ordered sequences of samples, and their corresponding sequences of labels are, respectively,* (●, ◎, X) *and* (■, ■, ◆)*, which are also ordered.*

- *Observe that the central point,* Lx_0*, may or may not belong to a chain. But when* Lx_0 *does not belong to a chain candidate the label* y_0 *is still used to compute if the sequence of labels is ordered. For example, although the sequence* (Lx_2, Lx_4, Lx_5) *is an ordered*

sequence of samples and its corresponding sequence of labels, $(y_2, y_4, y_5) = (\times, \blacksquare, \blacklozenge)$, is decreasing, it is not an ordered sequence of labels under our definition, since $y_0 \not\geq y_2 \geq y_4 \geq y_5$. Therefore, this sequence is not a chain.

The reason why we do not consider this last sequence a chain under our definition is because, from the point of view of distance-based ordinal regression, this chain does not move away from the base class of the neighborhood, y_0, gradually. The first element, y_2, is higher than y_0, while the second, y_4, is lower than y_0. Since in an ordinal regression problem the true distances between the class labels are unknown, to consider a sequence as a chain we are only interested in whether it ascends or descends from the base class of the neighborhood, y_0 in this case. Therefore, even though y_0 does not belong to the sequence, we impose this restriction to the definition of chain.

3.2 Optimization

Once the concept of chain has been defined, the problem that CMOML optimizes is

$$\max_{L \in \mathbb{R}^{d' \times d}} \sum_{i=1}^{N} S_L(x_i). \tag{2}$$

Observe that the objective function only takes integer values and therefore, it is not differentiable or even continuous. As a result, it is necessary to make use of black-box non-differentiable optimization methods in order to maximize Eq. 2. We will discuss the optimization method used in our experimental analysis in Section 4.2.

In order to avoid the brute force counting of $S_L(x_i)$, which would be computationally expensive according to the inequality in Eq. 1, it is possible to obtain the value of $S_L(x_i)$ using a dynamic programming approach. We will assume that there are no points in $U_L(x_i)$ at the exactly same distance from x_i. If this is not the case, the uniqueness in the order of the sequences of samples, which is needed for the dynamic programming approach, is lost. However, we will show at the end of the section that when we find several samples at the same distance from x_i we can still apply this algorithm by following a *pessimistic optimization* approach.

First, let us consider the sequence of all the labels associated with all the samples in $\mathcal{U}_L(x_i)$, ordered by the distances of the corresponding samples to x_i; in other words, the sequence of length K_i, $u = (y_{\tau(1)}, \dots, y_{\tau(K_i)})$ so that $\mathcal{U}_L(x_i) = \{Lx_{\tau(1)}, \dots, Lx_{\tau(K_i)}\}$ and

$$\|Lx_{\tau(1)} - Lx_i\| < \|Lx_{\tau(2)} - Lx_i\| < \cdots < \|Lx_{\tau(K_i)} - Lx_i\|.$$

Observe that the inequality above holds when using the strict order ($<$) thanks to the assumption of unequal distances around x_i. We will focus on counting the ascending chains around x_i. The process for counting descending chains is similar. From the sequence u we construct a subsequence v containing only the items in u that are greater than y_i, that is,

$$v = (y_{\tau\delta(1)}, \dots, y_{\tau\delta(R)}),$$

where $R \leq K_i$ and $\delta : \{1, \dots, R\} \to \{1, \dots, K_i\}$ is a strictly increasing map so that $y_{\tau\delta(j)} \geq y_i$ for every $j \in \{1, \dots, R\}$, and δ is also surjective over the set $\{j \in \{1, \dots, K_i\} : y_{\tau(j)} \geq y_i\}$

Now, the problem of counting the ascending chains around x_i is reduced to finding all the increasing subsequences of length κ inside v, since v contains all the labels in the neighborhood of x_i which are greater than y_i, ordered by the distances to x_i. This new problem can be easily solved using dynamic programming, by iteratively filling a $\kappa \times R$ matrix, where the (l, m) entry represents the number of sequences of length l which end at the m-th item in v, and can be recovered from the entries in the row $l-1$. This process can be done in $O(\kappa R^2)$.

The process of counting descending chains is done by constructing v with the elements of u that are lower than y_i, and counting the decreasing sequences in the new v. Observe that the constant chains are counted twice, since they appear in both ascending and descending chains. So, to obtain the final number of chains $S_L(x_i)$, we have to add the increasing chains and decreasing chains, and subtract the number of constant chains, which can be easily obtained as $\binom{H}{\kappa}$, where $H = |\{j \in \{1, \dots, K_i\} : y_{\tau(j)} = y_i\}|$.

Example. *We return to the dataset of Example 3.1 to show how to construct the sequence needed for the dynamic programming algorithm. To facilitate the calculations, we will identify the classes in the dataset with integer numbers:* ◆ $= 0$, ■ $= 1$, ● $= 2$, ◎ $= 3$, X $= 4$. *Since the samples in* $U_L(x)$ *verify that* $\|Lx_j - Lx\| < \|Lx_{j+1} - Lx\|$ *for* $j = 1, \dots, 5$, *we obtain the sequence* u *by adding the labels from* y_0 *to* y_6, *that is,* $u = (2, 3, 4, 1, 1, 0, 2)$. *To count the ascending chains, we compute* v *by removing the elements that are lower than* y_0 *from* u, *obtaining* $v = (2, 3, 4, 2)$. *Now we can apply the dynamic programming algorithm, obtaining one single chain:* $(2, 3, 4)$, *associated with the samples* (Lx_0, Lx_1, Lx_2) *(the ascending chain obtained in Example 3.1). To count the descending chains we follow a similar process, removing the labels greater than* y_0 *from* u, *and obtaining* $v = (2, 1, 1, 0, 2)$. *Now we apply the dynamic programming algorithm, obtaining three descending chains:* $(2, 1, 1)$, $(2, 1, 0)$ *and* $(1, 1, 0)$. *The last one is the descending chain shown in Example 3.1.*

Finally, let us explain how to handle the case where there are examples at the same distance from x_i. Observe that, if the points that are at the same distance from x_i share the same label, they can be swapped in u without affecting the final value of $S_L(x_i)$, so it does not matter in what order they are added to u, since both positions will have the same class label.

For the general case, let us note, first of all, that this situation is dependent on L, and we can apply a small disturbance to it that would break this equality (unless the samples at the same distance to x_i and x_i are aligned)[2]. This means that when we have two different samples being mapped by L to points at the same distance to x_i, it is easy to find another map in an

[2] If x_q, x_r verify that $\|Lx_q - Lx_i\| = \|Lx_r - Lx_i\|$ and x_i, x_q, x_r are not aligned, then $x_q - x_i$ and $x_r - x_i$ are linearly independent, and therefore we can define a linear transformation that maps them to different length vectors, resulting in different distances to x_i. This transformation can be arbitrarily close to L, since we could construct such a transformation by defining the image of $x_q - x_i$ and $x_r - x_i$ as $\lambda L(x_q - x_i)$ and $L(x_r - x_i)$, respectively, with $1 - \varepsilon < \lambda < 1$, for any $\varepsilon > 0$.

If x_i, x_q and x_r are aligned and verifying $\|Lx_q - Lx_i\| = \|Lx_r - Lx_i\|$, then $x_q = x_r$ or x_i is the middle point between x_q and x_r. The first case may be caused by class noise, and is discussed in the text. For the second case, the pessimistic approach described in the text will cause the objective function not to count chains that contain

arbitrarily small neighborhood of L that sends them to different points at different distances to x_i.

Moreover, having two different samples from different classes mapped to points at the same distance to x_i is not a positive thing from the point of view of a distance-based classification. For example, consider a sample x_1, of class 1, for which its nearest neighbors (for the distance provided by L) are x_2 and x_3, with classes 2 and 3, respectively, and x_2 and x_3 are at the same distance to x_1. To classify x_1 using the information from x_2 and x_3, a distance-based classifier would consider that x_1 has the same probability to belong to classes 2 and 3. However, in an ordinal regression problem we would expect that class 2 had a greater probability, since it is closer to the real class of x_1 than class 3.

Since we are using a black-box optimizer we propose a pessimistic approach, in which in case of equality, the vector u is constructed by adding the farther classes first. Thus, the number of chains obtained by the optimizer will be lower than the real one, and therefore we will be forcing the optimizer to search for a distance that breaks the equality in the most suitable way. In the previous example, with x_2 and x_3 at the same distance to x_1, we would add to u the class 3 before the class 2. In this way, the objective function for L will get a lower number of chains than the objective function for the disturbances of L for which x_2 is closer to x_1 than x_3 (which will add to u the class 2 before the class 3). Therefore the optimizer will be guided to find distances in this second scenario, which is more appropriate for a distance-based ordinal regression.

In any case, the occurrence of equal distances in a dense space like $\mathbb{R}^d$ is a very unlikely case, even assuming the finite precision of the floating point variables. Note that the pessimistic approach also covers the case of duplicate samples with different labels, for which this approach also allows a unique ordering to be established when constructing u, although in this case the duplicates will always be mapped to equal points. In this case, preprocessing techniques to reduce label noise may be useful before the execution of the algorithm [41, 42].

To sum up, the core of CMOML consists in counting the number of chains for a given linear map matrix L, which can be done using the procedure described above. This evaluation will be repeated for different matrices by a black-box optimizer in order to find an optimum value of chains. The optimizer we have used in our experimental analysis will be discussed in Section 4.2.

3.3 Non-linear CMOML

Since learning a Mahalanobis distance is equivalent to learning a linear transformation, there are many problems where the inherent non-linearity of the data cannot be properly handled by these distances. To solve this problem, we can use kernel functions in order to be able to learn a linear transformation in a higher dimensional feature space generated by a non-linear mapping. This idea has been successfully applied to other distance metric learning algorithms [43, 44], as well as being a fundamental piece in the success of an algo-

opposing elements within the neighborhood. This is also desirable, since those opposing samples cannot be modified by a linear transformation.

rithm of the relevance of the support vector machines [45]. We can apply the kernel trick to CMOML to obtain its kernel version, KCMOML.

We consider a non-linear mapping $\phi : \mathbb{R}^d \to \mathcal{F}$, where $\mathcal{F}$ is a Hilbert space of high dimension (or even infinite), denoted as the *feature space*. To learn a distance given by $L : \mathcal{F} \to \mathbb{R}^d$ using CMOML in the feature space we only need to compute the distances between the data after sending them to $\mathcal{F}$. Then the ordered sequences can be obtained in the same way as in the linear case once the distances are known. Assuming that a kernel function $\mathcal{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ associated with ϕ is known, if we search for the components of L in the span of $\{\phi(x_1), \dots, \phi(x_N)\}$, we can express L in terms of $\mathcal{K}$ and a new matrix A, of dimension $d' \times N$ [21] as

$$L\phi(x) = A \begin{pmatrix} \mathcal{K}(x, x_1) \\ \dots \\ \mathcal{K}(x, x_N) \end{pmatrix}.$$

From this identity, we can compute the distances $\|L\phi(x_i) - L\phi(x_j)\|$, as well as applying L to any point in $\mathcal{F}$. Therefore, by optimizing the matrix A in order to find the maximum number of chains in the feature space, using again a black-box non-differentiable optimization method (which will be discussed in Section 4.2) we obtain the kernelized version of CMOML.

3.4 Complexity Analysis of CMOML and Comparison with LODML

In this section we will compare the computational time complexity required by the CMOML and LODML methods and their kernel versions.

The parameters that determine the time complexity of LODML are: the number of samples N, the number of features d, the number of target neighbors k, the neighborhood size ν, and the size of the set of triplets $m = |R|$ [26]. Both the target neighbors and the neighborhoods can be built at the first stage of the algorithm using a linear nearest neighbors search, which can be performed in $\mathcal{O}(kN^2 + dN^2)$ and $\mathcal{O}(\nu N^2 + dN^2)$, respectively. The second stage of the algorithm is the optimization procedure, which is an iterative projected gradient descent method consisting of two parts: the gradient computation and its projection onto the positive semidefinite cone. The gradient computation requires the calculation of outer products involving all the constraints in R, which, in the worst case, can be performed in $\mathcal{O}(md^2)$. However, in practice, only a few outer products are required to be computed, which are those corresponding to the constraints that switched their activation state between consecutive iterations of the gradient update. Finally, the projection of the obtained gradient onto the positive semidefinite cone is performed in $\mathcal{O}(d^3)$ and the metric update is performed in $\mathcal{O}(d^2)$, thus the overall time complexity of LODML is $\mathcal{O}(d^3 + md^2)$ per iteration [26].

In the case of CMOML, the parameters that determine its complexity are: the number of samples N, the number of features d, the sizes of the neighborhoods $\{K_i : i = 1, \dots, N\}$, the sequence length κ and the output dimension d'. For the worst case estimation, let us assume

that $d' = d$ and $K = \max\{K_i : i = 1, \dots, N\}$. The objective function of CMOML requires transforming the dataset using L, which can be performed in $\mathcal{O}(Nd^2)$, and the computation of the pairwise distances for the transformed dataset, which can be performed in $\mathcal{O}(N^2d)$. Then, computing the number of chains in each neighborhood requires first the neighborhood calculation, which can be performed in $\mathcal{O}(KN)$ since all the pairwise distances in the transformed data were already computed. Then, the sequences u and v can be built in $O(K)$ as shown in Section 3.2, and finally the counting operation can be performed in $\mathcal{O}(kK^2)$ (also shown in Section 3.2 and using that $R \leq K$ in any neighborhood). This is done for each $x_i \in \mathcal{X}$, resulting in an overall time complexity of the objective function of

$$\mathcal{O}(Nd^2 + N^2d + N(KN + K + \kappa K^2)) = \mathcal{O}(N(d^2 + \kappa K^2) + N^2(d + K)).$$

To the objective function cost of CMOML we have to add the overhead of the differential evolution steps, which consists of $\mathcal{O}(Pd^2)$ after every P evaluations of the objective function [46], where P is the population size.

With respect to the kernel versions, both methods include an initial kernel matrix computation, of size $N \times N$, which can be performed in $\mathcal{O}(N^2d)$. Then, all the dimension-dependent (d) complexities in LODML scale to samples-dependent (N) complexities in KODML due to the kernel trick, resulting in an overall complexity of $\mathcal{O}(N^3 + mN^3)$ for the gradient iterations of KODML. For KCMOML, since we still count the chains after applying the transformation, the output dimension d remains in this stage and only the transformation operation and the differential evolution overhead scale to N, resulting in an overall complexity of $\mathcal{O}(N(Nd + \kappa K^2) + N^2(d + K))$ for the objective function, and $\mathcal{O}(PN^2)$ for the differential evolution overhead.

In summary, we see that both methods run in cubic orders of complexity with respect to (N, d), which is common in most metric learning methods, since at a minimum they usually require operations such as matrix products or decompositions.

Within the cubic order, LODML seems to be a lighter algorithm, and its gradient computation depends only on d. However, CMOML has the advantage that the evaluations of the objective function can be highly parallelized, and a whole generation can be evaluated simultaneously. In contrast, LODML is inevitably iterative as every iteration depends on the completion of the previous one. Finally, KCMOML scales somewhat better than KODML since it still depends on d in many terms of its overall complexity, rather than N.

4 Experiments

In this section we describe the experiments we have developed with CMOML and KCMOML, and the results we have obtained. The results obtained show a high performance of CMOML with respect to the compared algorithms.

First we describe the experiments, datasets and parameters we have used to evaluate the performance of CMOML. Then, we perform a distance metric learning comparison framework that involves CMOML, the Euclidean distance and the state-of-the-art of distance metric

learning for ordinal regression. We compare these algorithms using always the same nearest neighbors classifier. We do this for the linear case and for the kernelized versions of the algorithms. We show the results and verify them using Bayesian statistical analysis. Finally, we show the position of CMOML with respect to the state-of-the-art for ordinal regression, beyond distance metric learning and nearest neighbors classifiers.

4.1 Experimental Framework

We have evaluated the distance learned by CMOML using a distance-based classifier. Since we will consider ordinal datasets in the experiments, the classifier used in the experiments is the median-vote k-neighbors classifier [39], which is the natural adaptation for the common majority-vote k-neighbors classifier to ordinal regression, as discussed in Section 2.2. We have used a number of neighbors $k = 7$. This number of neighbors was chosen due to the fact that starting from this value the median and the mode of the neighbor labels begin to differ significantly. For smaller values they will be very similar in most cases. We have compared our algorithm with the results that the same classifier obtains when using the Euclidean distance, and the distance learned by LODML.

The different distances used by the classifier will be evaluated by a stratified 5-fold cross validation, that is, a cross validation that preserves the original class proportions in each fold. We have used 36 datasets, 23 of them being real-world ordinal regression datasets. The list of datasets has been completed by adding equal-frequency discretized regression datasets. All these datasets are numeric, without missing values, and have been *min-max* normalized to the interval $[0, 1]$ prior to the execution of the experiments. The datasets, their dimensions and the sources from which they have been gathered are shown in Table 1.

Let $h : \mathbb{R}^d \to \mathcal{Y}$ be the hypothesis obtained by the classifier. To evaluate it, we have used the *concordance index* (C-Index), which is one of the most popular metrics for ordinal regression [49, 50]. C-Index measures the ratio between the number of ordered pairs in both true labels and predictions and the number of all comparable pairs, that is

$$C = \frac{\sum_{i,j\,:\,y_i<y_j} \left([\![h(x_i) < h(x_j)]\!] + \frac{1}{2} [\![h(x_i) = h(x_j)]\!] \right)}{\#\{y_i, y_j : y_i < y_j\}}$$

where $[\![\cdot]\!]$ denotes the indicator function, that takes the value 1 if the condition inside is satisfied and 0 otherwise. C-Index is not influenced by the numerical representation of the class labels and can be understood as a generalization of the area under the ROC curve [51] for the ordinal case.

To the results on the different datasets we add the average score obtained for each metric, and an average ranking that has been calculated by assigning integer values starting from 1, according to the relative quality of the different algorithms for each dataset.

For CMOML and KCMOML we have used a fixed neighborhood size $K_i = 20$ for every sample in the dataset, a sequence length $\varkappa = 7$, an output dimension $d' = d$ and a differential evolution optimizer that is described in detail in Section 4.2. For LODML and KODML

Dataset	# Samples	# Features	# Classes	Source
affairs	265	17	6	[47]
automobile	202	71	5	[10]
autoMPG8	392	7	5	[48]
auto-riskness	157	15	5	[47]
balance	625	4	3	[48]
boston-housing	506	13	5	[47]
car	1728	6	4	[48]
cement-strength	998	8	5	[47]
cleveland	297	13	5	[48]
conctact-lenses	25	6	3	[10]
eucalyptus	736	91	5	[10]
glass	213	9	6	[47]
newthyroid	215	5	3	[48]
pasture	36	25	3	[10]
squash-stored	52	26	3	[10]
squash-unstored	52	25	3	[10]
tae	151	54	3	[48]
winequality-red	1359	11	6	[10]
winsconsin-breast-ord	194	32	5	[47]
ERA	1000	4	9	[10]
ESL	482	4	7	[10]
LEV	1000	4	5	[10]
SWD	1000	10	4	[10]
baseball [Discretized]	337	16	5	[48]
dee [Discretized]	365	6	5	[48]
ele-1 [Discretized]	495	2	5	[48]
forestFires [Discretized]	517	12	5	[48]
machineCPU [Discretized]	209	6	5	[48]
pyrim [Discretized]	74	26	5	[10]
stock [Discretized]	950	9	5	[10]
abalone [Discretized]	4177	11	5	[10]
bank1 [Discretized]	8192	8	5	[10]
bank2 [Discretized]	8192	32	5	[10]
computer1 [Discretized]	8192	12	5	[10]
computer2 [Discretized]	8192	21	5	[10]
calhousing [Discretized]	20640	8	5	[10]

Table 1: Datasets used in the experiments.

we have used a number of 7 target neighbors. Target neighbors are set, for each sample, as the nearest same-class neighbors to the sample for the Euclidean distance. Both the sequence length in CMOML and KCMOML and the number of target neighbors in LODML and KODML are set in order to match the number of nearest neighbors used in the subsequent k-NN classification.

Given that the purpose of this study is to draw a fair comparison between the algorithms and assess their robustness in a common environment with multiple datasets, we have not included a tuning step to maximize any particular performance metric.

The implementations of LODML, CMOML and their kernel versions used in this experimental analysis can be found in our Python library, `pydml` [52].

4.2 Black-box optimizer in CMOML and KCMOML

In order to run the CMOML and KCMOML algorithms it is necessary to establish some kind of black-box optimizer that can evaluate its objective function. In this experimental analysis we have used a differential evolution [53] algorithm. This evolutionary model is updated by taking differences between individuals in order to find a global optimum. Since the function to be optimized is not of high complexity, we have opted for this standard model of differential evolution, in both CMOML and KCMOML. The linear map matrices that define the distances are codified as a real vector of dimension $d' \times d$ that contains all the entries in the matrix added by rows (in the kernel version the matrix A is codified in the same way, but in this case we obtain a $d' \times N$ vector). We have taken the differential evolution implementation available in Scikit-Learn[3] and the parameters we have used are:

- A population size of 100, initialized using the latin hypercube strategy, in order to maximize the coverage of the parameter space [54].
- The algorithm has been executed during 150 generations.
- The strategy used to generate new candidates has been *best1bin*, in which the best solution is updated, depending on the recombination probability, with the difference of a pair of individuals taken randomly.
- The decision of updating each of the parameters is made using a binomial distribution with a recombination probability of 0.7. The differential weight that determines how much a parameter is updated in case of recombination is taken randomly, for each generation, from the interval $[0.5, 1[$. When the new individual is constructed, its fitness is calculated using Eq. 2 (after rewriting it as a matrix) and it replaces the original individual if the new fitness is higher.

[3] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html`

4.3 Results for the Linear Case

Table 2 shows the C-Index results obtained by the algorithms. According to these results, we see a clear dominance of CMOML over the other algorithms. This shows once again the main ability of our algorithm, since a high C-Index indicates that the data are ordered in both the input and the output space.

In order to evaluate to what extent CMOML outperforms the compared algorithms and vice-versa, we have conducted a series of Bayesian statistical tests. We have prepared several Bayesian sign tests [55] that compare CMOML with each of the other algorithms. These tests take into account the differences between the C-Index scores obtained from the two algorithms being compared, assuming that their prior distribution is a Dirichlet Process [56], defined by a prior strength $s = 1$ and a prior pseudo-observation $z_0 = 0$. After reading the score observations obtained for each dataset, the tests produce a posterior distribution. This distribution provides us with the probabilities that each of the compared algorithms will outperform the other. The tests also introduce a *region of practically equivalent (rope)* performance, where it is assumed that neither of the algorithms is better than the other. We have designated the rope region as the one where the score differences are in the interval $[-0.01, 0.01]$. In summary, from the posterior distribution we obtain three probabilities: the probability that the first algorithm will outperform the second, the probability that the second algorithm will outperform the first, and the probability that both algorithms will have an equivalent performance. The distribution can be displayed in a simplex plot for a sample of the posterior distribution, where a greater tendency of the points towards one of the regions will represent a greater probability.

To carry out the Bayesian sign tests we have used the R package `rNPBST` [57]. Figure 3 shows the results of the comparisons using the C-Index metric.

By examining the results of the tables and the corresponding Bayesian analyses, we can draw several conclusions. The Bayesian analysis shows us that there is a very low probability that Euclidean distance will outperform CMOML, while CMOML has a greater chance of outperforming Euclidean distance, although there is also a remarkable probability that both distances have an equivalent performance. In the comparison between LODML and CMOML we observe again that CMOML has a very high probability of outperforming LODML, with low probabilities for the reciprocal and rope cases.

In general, according to the results observed with this ordinal regression metric, we can conclude that CMOML could be an interesting alternative when looking for distances aimed at ordinal regression problems.

4.4 Results for the Non-Linear Case

Finally, we will show the results of the experiments with the kernelized versions of CMOML and LODML. We have evaluated KCMOML and KODML over the datasets previously described in Table 1, using the same classifier, validation strategies, parameters and metrics. Since both kernel versions learn linear transformations that scale quadratically with the

	Euclidean	LODML	CMOML
affairs	**0.569363**	0.565897	0.554959
automobile	0.808623	**0.880245**	0.815365
autoMPG8	**0.927234**	0.895400	0.917576
auto-riskness	0.614520	0.627593	**0.651275**
balance	0.925787	0.960857	**0.976474**
boston-housing	0.795998	0.825344	**0.846487**
car	0.972751	**0.980798**	0.978543
cement-strength	0.738981	0.734096	**0.844991**
cleveland	0.783723	**0.788992**	0.767258
contact-lenses	0.700000	0.700000	**0.828571**
eucalyptus	0.823725	**0.881438**	0.804636
glass	0.774106	0.752447	**0.794482**
newthyroid	0.889352	**0.945833**	0.943981
pasture	**0.867593**	0.801852	0.845370
squash-stored	0.665731	0.721389	**0.773538**
squash-unstored	0.721429	0.772857	**0.803571**
tae	**0.686458**	0.636642	0.677773
winequality-red	0.701366	0.649910	**0.703374**
wisconsin-breast-ord	**0.651560**	0.621095	0.589265
ERA	0.689033	**0.693902**	0.680311
ESL	0.912828	0.906105	**0.917878**
LEV	0.808138	0.792477	**0.810350**
SWD	**0.744839**	0.749787	0.748572
baseball	0.863565	0.766930	**0.877246**
dee	0.875449	0.846101	**0.879377**
ele-1	**0.878193**	0.855523	0.877938
forestFires	0.519372	0.519587	**0.540831**
machineCPU	0.854202	0.860928	**0.871658**
pyrim	0.764274	**0.805812**	0.800085
stock	0.962147	0.688421	**0.962486**
abalone	0.802877	0.724529	**0.805626**
bank1	0.773963	0.942145	**0.949363**
bank2	0.587407	0.770863	**0.784161**
computer1	0.903586	0.784361	**0.910165**
computer2	0.906698	0.855022	**0.909880**
calhousing	0.866630	0.778105	**0.910898**
AVG SCORE	0.786986	0.780091	**0.815398**
AVG RANK	2.229730	2.256757	**1.513514**

Table 2: C-Index score for the linear version of the algorithms in each dataset.

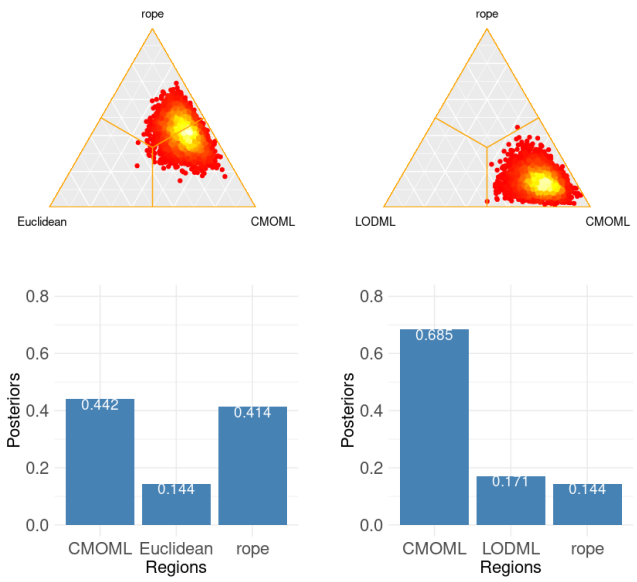


Figure 3: Bayesian sign test results for the comparison between CMOML with Euclidean distance and LODML using C-Index. Simplex diagrams and posterior distributions are shown.

number of samples, which can be computationally expensive for large datasets [26], we have set a time limit of one week for the kernel experiments. We have evaluated both algorithms using two of the most popular kernels, namely:

- The *polynomial kernel*, $\mathcal{K}(x, x') = \gamma\langle x, x'\rangle^p$. We have used $p = 2$, thus we have obtained a quadratic transformation of the data.
- The *radial basis function kernel*, or RBF. It is defined as $\mathcal{K}(x, x') = \exp(-\gamma\|x - x'\|^2)$. The image of the non-linear mapping associated with this kernel is an infinite-dimensional space.

The value γ in both kernels has been tuned by cross-validation using the set of values $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100, 1000\}$.

The results of the kernelized versions using the C-Index score are shown in Table 3. The symbol (-) in a cell of the table indicates that the corresponding algorithm has exceeded the established time limit. Observing these results we can see that the kernel versions are able to obtain better results than the linear versions in most of the datasets. The improvement observed for KODML with respect to LODML is higher than the improvement of KCMOML with respect to CMOML, but KCMOML still obtains better average results, with the RBF kernel obtaining the best average scores and the polynomial kernel obtaining the best average rankings. Analyzing the results of the Bayesian tests[4] from Figure 4 we can see there is pretty level playing field between the two algorithms when using the polynomial kernel,

with a very high probability that the algorithms will have equivalent performances, while with the RBF kernel the probability distribution in the three regions is very even, although slightly biased towards KCMOML. This tells us that KCMOML has a slightly higher probability of performing better than KODML, although a similar probability for the KODML region implies that the algorithms are able to complement each other well for different datasets. It is also interesting to observe that KCMOML with the RBF kernel outstands in most of the real-world ordinal regression datasets, while its performance on the discretized regression datasets slightly worsens. Finally, it should be noted that KCMOML only times out in 2 of the large datasets used in the experiments, while KODML times out in 5 of them. This confirms what the complexity analysis showed about KCMOML scaling better than KODML with respect to the number of samples.

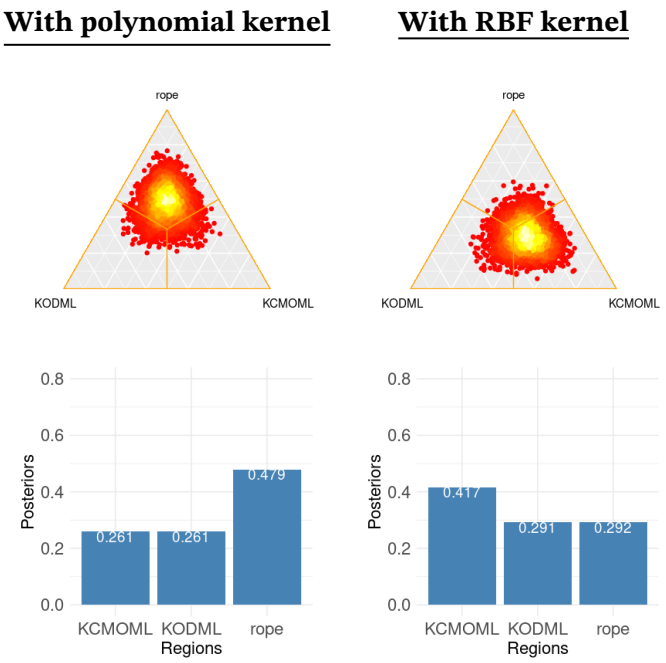


Figure 4: Bayesian sign test results for the comparison between KCMOML and KODML, with polynomial and RBF kernels, using C-Index. Simplex diagrams and posterior distributions are shown.

4.5 Comparison with State-of-the-art Methods for Ordinal Regression

Since CMOML has proven to be an outstanding algorithm within the family of distance metric learning algorithms for ordinal regression, in this section we compare the proposed algorithm to other state-of-the-art algorithms for ordinal regression. The experiments were

[4]In order to conduct a fair comparison between the two methods, the average score, the average ranking and the Bayesian test results exclude the datasets *bank1*, *bank2*, *computer1*, *computer2* and *calhousing* in the kernel experiments.

	KODML		KCMOML	
	POLY-2	RBF	POLY-2	RBF
affairs	0.586422	0.545643	0.567910	**0.601115**
automobile	0.814357	0.802392	**0.862178**	0.851728
autoMPG8	0.927905	0.908938	0.921637	**0.928632**
auto-riskness	0.664561	**0.708308**	0.660411	0.671843
balance	0.965443	0.986929	**0.993742**	0.991133
boston-housing	**0.834244**	0.814329	0.814222	0.807763
car	0.978700	0.973873	0.984611	**0.985772**
cement-strength	0.784070	0.760296	**0.818427**	0.806169
cleveland	0.797910	**0.814295**	0.778811	0.785053
contact-lenses	0.857143	0.857143	0.871429	**0.885714**
eucalyptus	**0.874315**	0.851291	0.827302	0.793716
glass	0.774331	0.774331	0.797583	**0.802359**
newthyroid	0.948843	0.949769	0.943981	**0.966435**
pasture	0.861111	0.846296	**0.867593**	0.831481
squash-stored	0.753582	0.718582	0.771637	**0.858450**
squash-unstored	0.780714	**0.815000**	0.757857	**0.815000**
tae	0.700905	**0.725865**	0.675628	0.700214
winequality-red	0.703518	0.681436	**0.705966**	0.702759
wisconsin-breast-ord	0.656351	**0.659343**	0.616383	0.603888
ERA	0.702709	0.703945	**0.706104**	0.702813
ESL	0.923134	0.920086	**0.926032**	0.920399
LEV	0.813103	0.808132	**0.813145**	0.810783
SWD	0.755780	**0.757707**	0.752039	0.756966
baseball	0.863850	0.805774	**0.866594**	0.865433
dee	0.882748	0.861711	**0.889650**	0.888016
ele-1	**0.881161**	0.877794	0.879256	0.870785
forestFires	0.538324	**0.541348**	0.520120	0.517618
machineCPU	**0.877991**	0.864206	0.872821	0.854135
pyrim	0.779829	0.815556	**0.837949**	0.782308
stock	**0.963650**	0.955048	0.958975	0.961427
abalone	0.731883	0.736535	**0.768793**	0.756074
bank1	-	-	0.844700	**0.884079**
bank2	-	-	-	-
computer1	-	-	0.888985	**0.899613**
computer2	-	-	0.865846	**0.913115**
calhousing	-	-	-	-
AVG SCORE[4]	0.805760	0.801351	0.807379	**0.808903**
AVG RANK[4]	2.532778	2.859521	**2.217482**	2.390218

Table 3: C-Index score for the kernel version of the algorithms in each dataset.

performed using the two methods that obtained the best results in the experimental analysis of [10], namely, SVOREX [13] and REDSVM [32]. These two algorithms are adaptations of support vector machines for ordinal regression. The first adds explicit constraints concerning adjacent classes for threshold determination, while the second reduces the ordinal regression problem to be applied to binary SVM classifiers.

In addition to these SVM variants we have included two more recent proposals for ordinal regression in our comparison. The first is an adaptation of the extreme learning machines, *kernel extreme learning machine for ordinal regression* (KELMOR) [16]. The second is an extension of the Bayesian network classifiers [17]. This extension learns a Bayesian network that jointly maximizes accuracy and mutual information, in order to better adapt to imbalanced and ordinal problems. We will refer to this algorithm as EBNC (*extended Bayesian network classifier*).

Both SVMs and KELMOR have been used with a Gaussian kernel and an adjusting parameter $C = 10$. For the Bayesian network classifier, the features were discretized to a maximum of ten values per attribute, using equal-length bins, so that they could be handled by the network. CMOML has been used with the same settings as in Section 4.3. As in the previous experiments, we have carried out a stratified 5-fold cross validation. We have taken the implementations of the SVMs from [58] and the code of EBNC provided by [17]. We also provide an implementation of KELMOR[5]. The C-Index scores obtained by CMOML and each of the state-of-the-art classifiers are shown in Table 4.

Looking at the results, we can observe that CMOML is slightly behind the SVMs in terms of rankings. However, although CMOML only ranks first in 6 of the datasets in C-Index, it achieves outstanding wins in datasets such as *newthyroid*, *glass* or *calhousing*, with convincing wins in the rest of its top positions as well. Moreover, in most of the cases where CMOML performs worse than the other algorithms, its results are still competitive with the results of the rest of algorithms. This translates into the best average C-Index.

In summary, CMOML has the ability to excel in several ordinal datasets where other state-of-the-art methods do not perform well, in addition to having decent overall performance. Finally, as we will see in the next section, CMOML stands out from the compared methods in terms of explainability, regarding case-based reasoning, dimensionality reduction and visualization. Both support vector machines and extreme learning machines are considered by design complex black-box learning algorithms [19, 59]. Their opaque structure turns them into undesirable algorithms for high-risk automated tasks. In contrast, the transparency of the nearest neighbors-based algorithms and the information provided by the neighbors themselves make them much more useful in these tasks [30]. We will see in the next section that the distance learned by CMOML, besides considerably improving the performance of the k-NN, makes the interpretable information produced by the classifier much more intuitive.

[5] `https://github.com/jlsuarezdiaz/KELMOR`

	REDSVM	SVOREX	EBNC	KELMOR	CMOML
affairs	0.543764	0.479077	0.544441	**0.570416**	0.554959
automobile	**0.853380**	0.759912	0.776466	0.838471	0.815365
autoMPG8	0.921465	**0.933516**	0.915756	0.918324	0.917576
auto-riskness	0.696078	**0.763215**	0.740496	0.696718	0.651275
balance	0.988387	**0.994224**	0.930498	0.949678	0.976474
boston-housing	0.838319	0.838910	0.763829	**0.851915**	0.846487
car	0.956863	**0.982058**	0.974778	0.952840	0.978543
cement-strength	0.836725	**0.864907**	0.702598	0.799276	0.844991
cleveland	0.820070	0.794660	0.733414	**0.835190**	0.767258
contact-lenses	0.700000	0.685714	0.614286	**0.885714**	0.828571
eucalyptus	**0.882360**	0.854718	0.853683	0.868132	0.804636
glass	0.717674	0.773195	0.642967	0.734655	**0.794482**
newthyroid	0.750463	0.857407	0.833796	0.776157	**0.943981**
pasture	0.827778	0.757407	0.846296	**0.854630**	0.845370
squash-stored	0.777310	0.811784	**0.830044**	0.766608	0.773538
squash-unstored	0.761429	0.775000	0.710000	0.775714	**0.803571**
tae	0.604379	**0.684285**	0.636560	0.594199	0.677773
winequality-red	0.723094	**0.730985**	0.726721	0.720977	0.703374
wisconsin-breast-ord	**0.615969**	0.599485	0.516718	0.604256	0.589265
ERA	0.710939	0.680576	0.692489	**0.720854**	0.680311
ESL	0.929687	**0.932802**	0.908091	0.929938	0.917878
LEV	0.811428	**0.822995**	0.801398	0.809583	0.810350
SWD	0.744378	0.761199	**0.780408**	0.735707	0.748572
baseball	**0.881180**	0.856130	0.830042	0.875346	0.877246
dee	**0.900075**	0.894234	0.868829	0.889131	0.879377
ele-1	**0.885381**	0.884788	0.848548	0.870766	0.877938
forestFires	0.515009	0.512263	0.499984	0.523941	**0.540831**
machineCPU	**0.885147**	0.882376	0.777610	0.865265	0.871658
pyrim	**0.852821**	0.732393	0.729316	0.792393	0.800085
stock	0.922424	0.959384	0.940706	0.924979	**0.962486**
abalone	0.813377	**0.823557**	0.769140	0.813790	0.805626
bank1	**0.953372**	0.952471	0.714695	0.923420	0.949363
bank2	**0.824870**	0.773617	0.708871	0.801277	0.784161
computer1	0.899223	**0.910422**	0.808520	0.899727	0.910165
computer2	0.910846	**0.927012**	0.880130	0.914840	0.909880
calhousing	0.862286	0.874078	0.801631	0.857212	**0.910898**
AVG SCORE	0.808832	0.808910	0.768160	0.809501	**0.815398**
AVG RANK	2.729730	**2.405405**	4.162162	2.891892	2.810811

Table 4: C-Index score for the state-of-the-art methods and CMOML in each dataset.

5 Nearest Neighbors and Metric Learning for an Explainable Learning Process

In this section we will explore the explainability possibilities offered by CMOML when it is applied together with the nearest neighbors classifier. *Explainable artificial intelligence (XAI)* [19, 20] has recently gained new relevance as a research topic as a consequence of the growing need for transparency and interpretability in the large amount of highly complex models, such as ensembles or deep neural networks, that currently dominate machine learning.

We saw in the previous section that CMOML obtains results close to those of the state-of-the-art for ordinal regression. The purpose of this section is to show that, in addition to the foregoing, CMOML has the advantage over the state-of-the-art algorithms in that, since it can be used in combination with a distance-based algorithm such as the nearest neighbors classifier, it can benefit from the strengths that k-NN provides in terms of explainability, and even polish them in several ways, as we will see throughout the section.

The nearest neighbors classifier, like most similarity-based classifiers, can be interpreted in terms of *case-based reasoning* [30]. The k-NN makes its decisions based on the similarity with previous experiences (the k nearest neighbors in the learned training set). Therefore, it is possible to analyze these experiences to decide to what extent the decision made by the algorithm can be trusted. This is very similar to human decision making, which often relies on previous experiences. In addition, when our data is low-dimensional it is possible to visualize why the k-NN has decided to make a certain choice, thanks to the simplicity of the "nearest neighbor" concept.

If we want to classify a new sample using the k-NN and we fit an appropriate distance to our training data, the nearest neighbors of the sample may change and become even more similar to the sample. In this way, these neighbors will be more informative when interpreting a decision. This is possible with distance metric learning. In particular, while the CMOML optimization algorithm improves the layout of the data to perform better under ordinal regression metrics, it also modifies nearby instances to be more related to the sample to be predicted. In addition, we can pick the dimension to which we want CMOML to project the data.

Therefore, in terms of explainability, CMOML improves the traditional k-NN in two aspects:

- **Comprehensibility.** The knowledge the k-NN with CMOML learns can be represented by the nearest neighbors, and these neighbors will be much more informative than the neighbors the Euclidean distance would obtain.
- **Understandability.** The ability to reduce the dimensionality of CMOML allows data to be represented in spaces of lower complexity where it is easier to understand why the classifier makes a certain decision.

We will test these CMOML capabilities in two different datasets: *balance* and *newthyroid*. In these datasets, CMOML outperforms the Euclidean distance and also obtains good results

compared to the state-of-the-art algorithms, as shown in Table 4. CMOML clearly outperforms all the algorithms in *newthyroid* and it is very close to the SVMs in *balance*. In this last case, the gains in interpretability that we are going to show may compensate the minimal loss in the ordinal metrics.

5.1 Knowledge representation with CMOML

Here we will analyze how the distance learned by CMOML improves knowledge representation for a case-based reasoning, with the neighbors obtained by the k-NN. To do this, we follow the outline in Figure 5. In short, once the distance is learned, we use it to transform the new sample that is to be classified. In the transformed space we retrieve its nearest neighbors. With them, on the one hand, the classifier makes its prediction, aggregating the classes of the neighbors (for the ordinal case, we use the median class here again). On the other hand, we retrieve the neighbors in the original space and visualize them together with the case to be predicted.

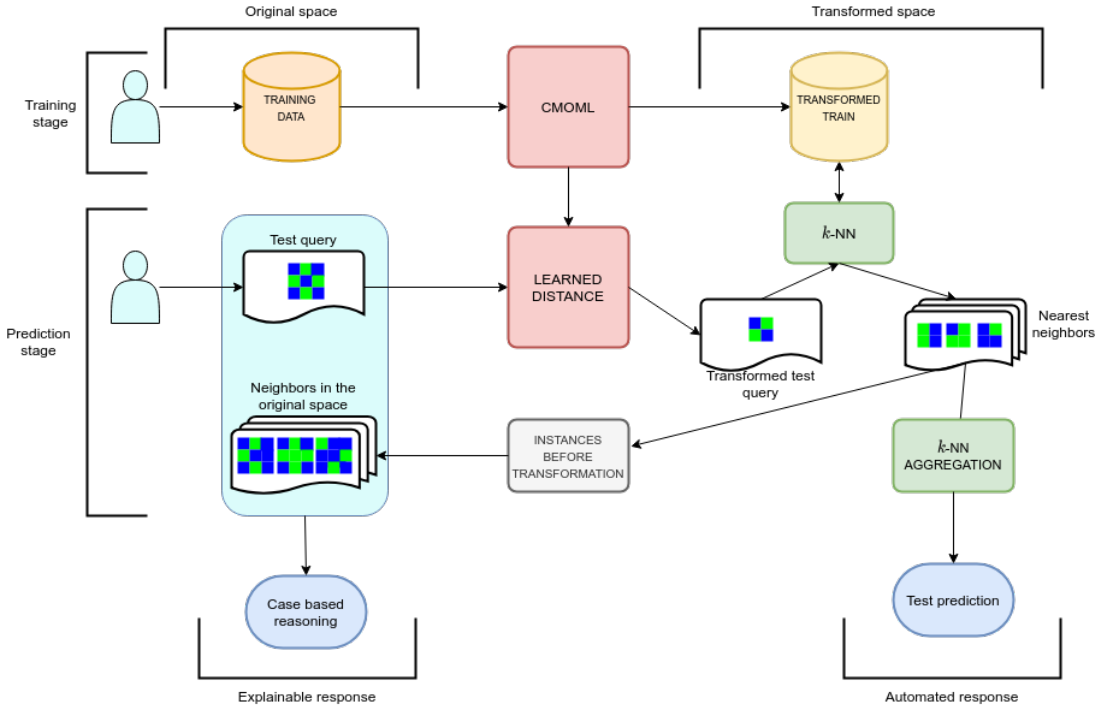


Figure 5: A case-based reasoning approach with nearest neighbors and CMOML.

Below we perform this procedure with several examples in *balance* and *newthyroid*.

5.1.1 Balance dataset analysis

Balance is a dataset whose examples represent two objects placed at the sides of a scale. Its attributes consist of the weight and the position of the left item and the weight and the

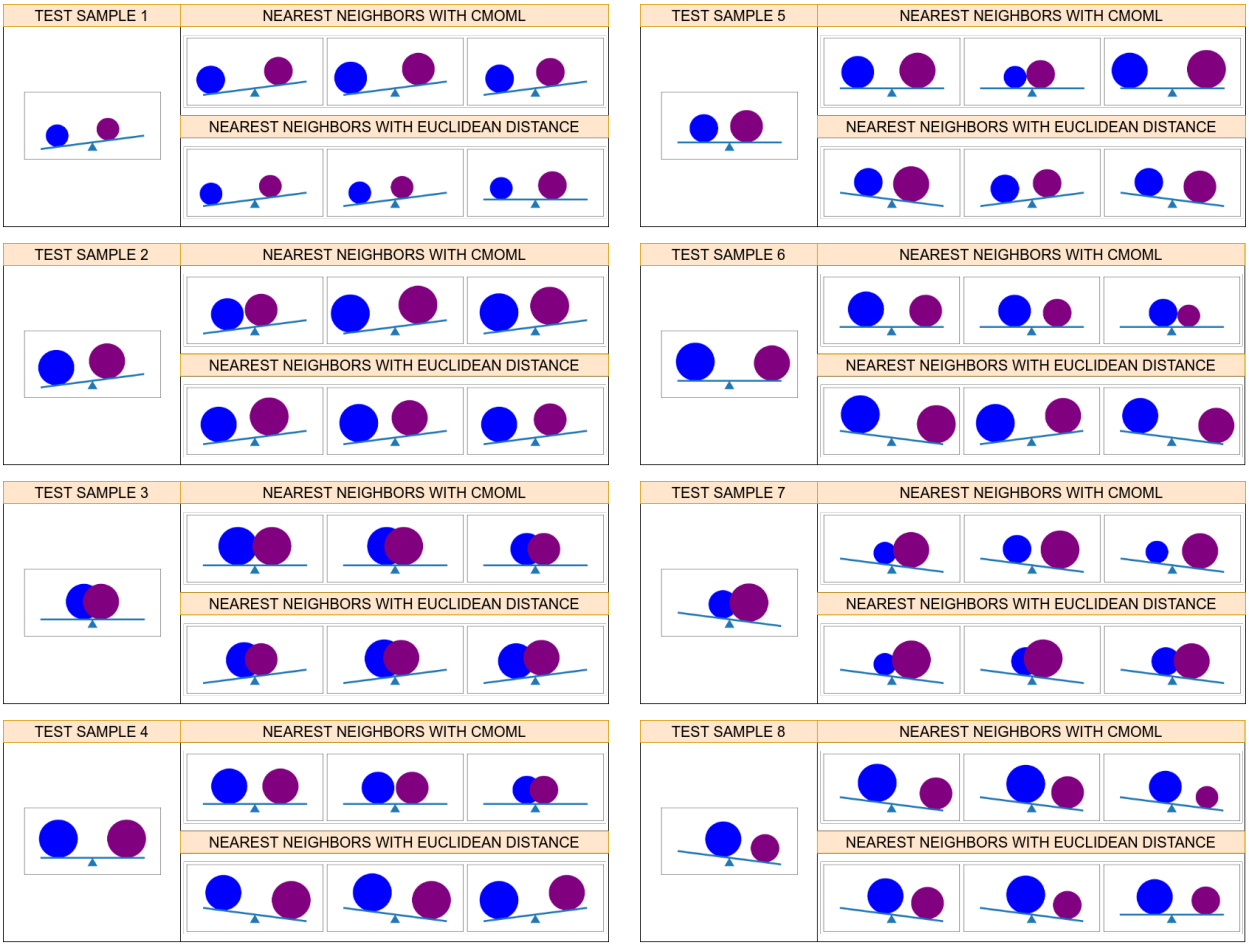


Figure 6: Nearest neighbors of some test samples in `balance` using CMOML and Euclidean distance.

position of the right item. The goal is to predict whether the scale tilts to one side (*right* or *left*) or whether it stays in balance. Due to the continuity of the scale movement it is logical to assume the relation of order in the output variable $left < balance < right$. This dataset is very useful to help visualize how CMOML improves the neighbors that the k-NN retrieves. Indeed, we will see that these neighbors are very similar to what a human being would consider, if prompted to decide how the scale will move based on previous experiences.

We train CMOML with 80 % of the dataset and use the rest for testing. For the test data, we retrieve the 3 nearest neighbors obtained by CMOML, and we also retrieve the nearest neighbors that the Euclidean distance would obtain. Figure 6 shows the results for some of the test samples.

From the results obtained we can draw several conclusions. On the one hand, for the test data that are not in balance, the neighbors that the two distances obtain are often same-

class neighbors, especially for CMOML. This is reasonable due to the geometric properties of the dataset. Observe that the true labels are determined by the sign of `left_weight * left_distance - right_weight * right_distance`, so the *left* and *right* classes are determined by half-spaces. Even so, we can see that the neighbors obtained by CMOML are more intuitive than those obtained by the Euclidean distance. For example, we can see, for the test sample 1, that the neighbors that CMOML obtains are always objects of the same weight, which also keep ratios between the distances to the center of the scale. Presumably, if we, as humans, had to decide where the scale would tilt to based on the same previous experiences, we would have chosen those very same neighbors. With the Euclidean distance this does not happen so clearly.

On the other hand, for the test data that are in balance, the Euclidean distance often tends to choose neighbors that tilt to one of the sides. This is due to the fact that this distance is not optimized and the *balance* class is a hyperplane between the other two classes, so as soon as a training sample of the *left* or *right* class is close to this border it can easily interfere with the neighbors. Instead, the neighbors that CMOML obtains are mostly in the *balance* position. And not only that, but again they are much more intuitive. Some of the clearest cases are the test samples 3 and 4. There, both objects weigh the same and are at the same distance from the center of the scale. All their neighbors share this property. Therefore, the answer is clear when analyzing why these neighbors have been chosen.

5.1.2 Newthyroid dataset analysis

Newthyroid is a dataset whose input variables are the measurements of certain hormones in the human body: `T3 resin`, `thyroxin`, `triiodothyronine`, `thyroidstimulating` and `TSH`. The goal is to predict, with these measures, whether the individual being evaluated suffers from hyperthyroidism, hypothyroidism or is healthy. As both disorders can be considered opposed, it is assumed that the ordinal relationship of the output variable is $hypothyroidism < normal < hyperthyroidism$.

Once again, we train CMOML with 80 % of the dataset, use the rest for testing, and retrieve the 3 nearest neighbors obtained by CMOML and by the Euclidean distance for the test data. In Figure 7 we show the neighbors obtained for some of the most remarkable cases. Here, we represent an instance by a heatmap, where each cell is each of the input variables in the data set, in the same order as described above. These cells range from the lowest possible value for the attribute (green) to its highest possible value (red). We also highlight the border of the heatmaps to specify the true labels of each instance. Blue borders represent instances of class *hypothyroidism*, green borders represent instances of class *normal* and red borders represent instances of class *hyperthyroidism*.

From the properties we have observed we can conclude several facts. First of all, it is very common to always find neighbors of the class *normal* for the test samples in the *normal* class. This is reasonable, since the normal class is the most frequent and quite dense, as we will see in the next section. It is also the least important class of the problem, since, as with most medical-related problems, false positives in disease diagnosis are not as serious

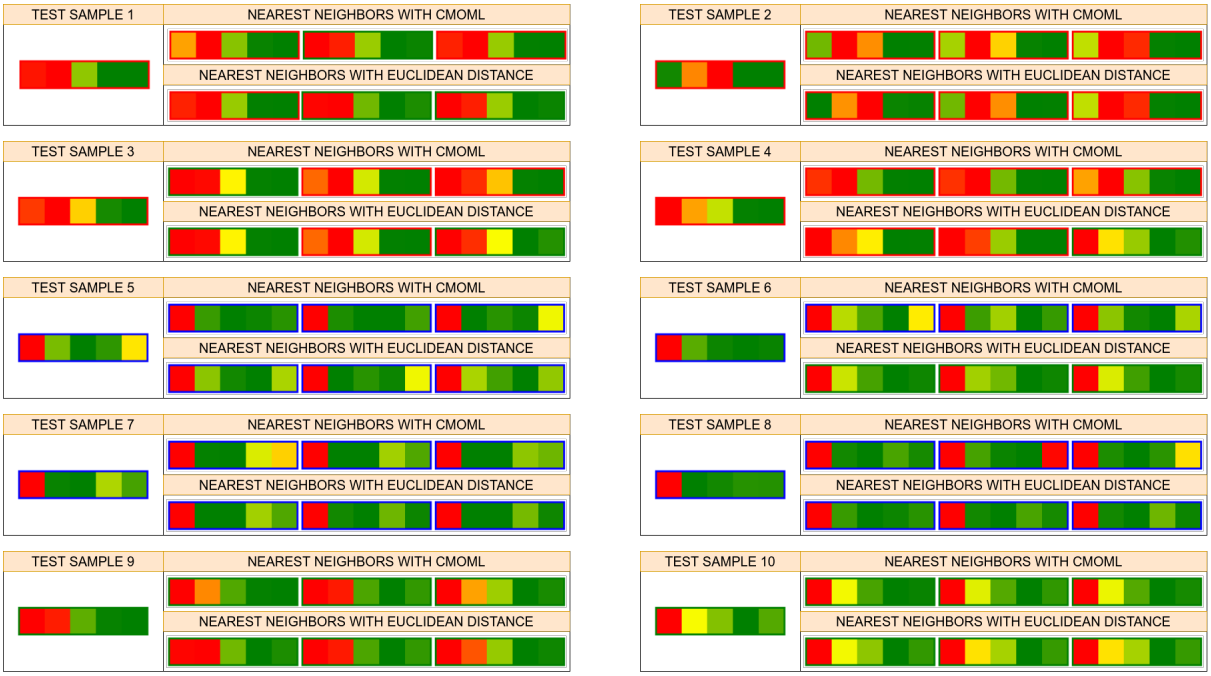


Figure 7: Nearest neighbors of some test samples in `newthyroid` using CMOML and Euclidean distance.

as false negatives. That is why in Figure 7 we choose to focus on the possible positives (both *hypothyroidism* and *hyperthyroidism*) to perform a case-based reasoning, taking only the samples 9 and 10 from the *normal* class.

Focusing now on the samples with *hyperthyroidism* (samples 1-4) and *hypothyroidism* (samples 5-8), we can see that the neighbors obtained by CMOML are much more label-accurate than the neighbors obtained by the Euclidean distance. In fact, in some cases like test samples 3 and 6, the k-NN with the Euclidean distance would misclassify the samples, since the median class in these cases would be *normal*. CMOML can still classify these samples correctly because most of the neighbors it obtains are from the correct class.

Finally, we analyze how the distance learned by CMOML influences the values of the features of the nearest neighbors. We can observe that, while the Euclidean distance has the limitation of only providing neighbors whose features are all very similar one-by-one to the features of the test sample, CMOML can go a step further and provide neighbors with more distinguishing properties. For example, it is known that low values of `thyroxin` combined with very low or high values of `TSH` are usually a symptom of *hypothyroidism*. For the test sample 6 we observe a low value of `TSH` and a medium-low value of `thyroxin`. The neighbors obtained by the Euclidean distance show `thyroxin` values that are not low enough and, additionally, they are tagged as *normal*. However, CMOML provides neighbors with lower values of `thyroxin`, and values of `TSH` that are both low and high. That is, the algorithm is

discovering that both low and high values of TSH influence *hypothyroidism*, and finds a new similarity between the data that the Euclidean distance overlooks.

5.2 Dimensionality reduction and visualization

To conclude, we will analyze how the dimensionality reduction applied by CMOML allows for the visualization of the transformed data, so that the decision made by the nearest neighbors classifier can be understood. To do this, we learn a distance with CMOML for each dataset and impose a dimension of 3 and 2 in the transformed space. In this way, the algorithm will obtain a projection of the data in the three- and bi-dimensional spaces that will be adequate from an ordinal perspective, as shown throughout the paper. With this dimensionality reduction, the results of the cross validation over *balance* decrease by less than a hundredth for the 3D projection, using C-Index, and two hundredths more in the 2D projection. In *newthyroid* the projections maintain the same results as at maximum dimension, as it is shown in Table 5. Therefore, we can conclude that the reduction does not significantly affect the quality of the data, and thus the transformed data can be used to extract human-understandable information.

	C-INDEX	
Dimension	balance	newthyroid
MAX	0.976474	0.943981
3	0.973122	0.943981
2	0.970433	0.943981

Table 5: C-Index scores with CMOML in balance and newthyroid after applying a dimensionality reduction

In Figures 8 and 9 we show how the data from *balance* and *newthyroid*, respectively, look like when projected on three and two dimensions. In the 2D-projection we also include the test samples from Figures 6 and 7, to facilitate the understanding of the choices the classifier makes. The test samples are marked with red numbers. Each number corresponds to the sample number they had in the aforementioned figures.

By analyzing the 3D and 2D projections of *balance* we can see why the *left* and *right* classes are so easy to classify and why the *balance* class presents a major difficulty. Indeed, the latter class acts as a hyperplane that separates the other classes. Therefore, if the training set is not very populated in this class, it is easy for neighbors of the *left* and *right* classes to decrease the quality of the classification of true *balance* instances. In any case, the projections learned by CMOML, especially at maximum dimension and to the 3D-space, have proven to be good enough on the central class. Moreover, on the 2D projection we can observe that, indeed, the test samples of Figure 6 of the class *balance* always fall on training samples of the class *balance*.

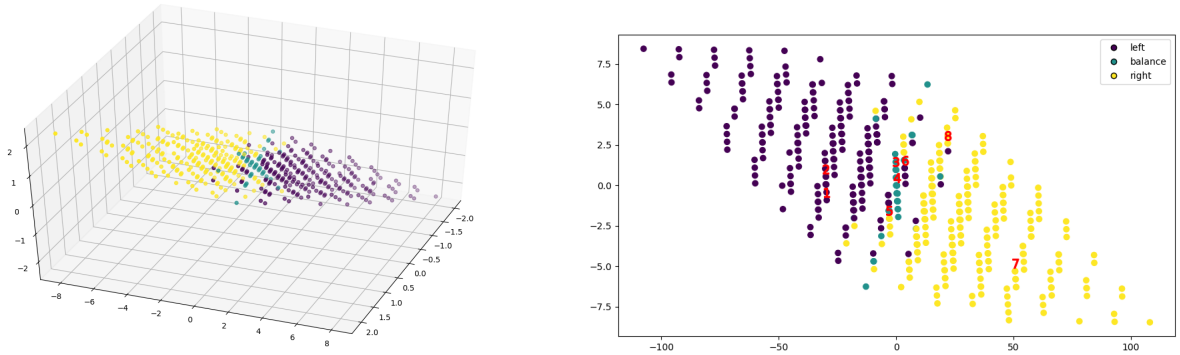


Figure 8: 3D and 2D projections of `balance` learned by CMOML. The 2D projection shows the position of the transformed test samples in Figure 6.

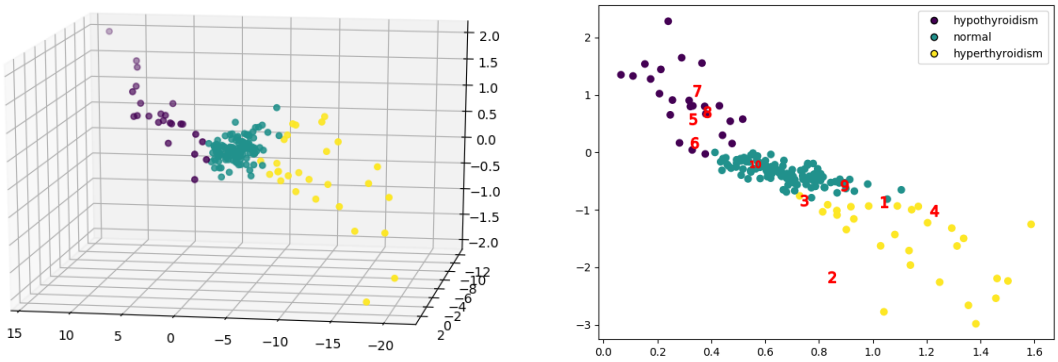


Figure 9: 3D and 2D projections of `newthyroid` learned by CMOML. The 2D projection shows the position of the transformed test samples in Figure 7.

If we now analyze the projections of *newthyroid*, we see that the *normal* class stands in the center between the two disorders, while the points of the other classes usually take values that shoot up in some dimension. In general, there is a fairly clear separation among the classes, with the exception of certain border points for which some doubts may arise when it comes to classification. This can be corroborated with the test samples from Figure 7 that are plotted in the 2D-projection: it is clear that most of them can be correctly classified using the scatter plot, while the test samples 1 and 3 can cause some more confusion, as can be seen in Figure 7.

It should be noted that the projections respect the ordinal principle by which CMOML is guided: that, as we move farther away from any sample, the classes will gradually become more different. This happens with the two datasets analyzed. Finally, we have to mention that 2D or 3D projections are not always possible with CMOML without a significant loss of information. In any case, a dimensionality reduction even to dimensions greater than 2 and 3 can be beneficial in terms of efficiency and noise reduction. In addition, it is always

possible to use other visualization methods in larger dimensions [60], which can perform better if we first apply CMOML properly.

6 Conclusions

In this paper we have developed a new distance metric learning algorithm for ordinal regression, following a new approach based on the optimization of ordered sequences. This approach is purely ordinal, since it makes an extensive use of the relative positions among the labels and it has proven to be effective in situations where other proposals on the same topic cannot operate properly.

According to how the proposal has been developed and how the results have supported it, we can conclude that CMOML is a promising algorithm with strong capabilities in numerous ordinal regression problems. The sequence based approach provides CMOML with a different and effective way of handling the ordinal regression problems, compared to the previous distance metric learning proposals in the subject. In the general context of ordinal regression, CMOML has also proven to be competitive and more explainable that the best performing methods, and it improves significantly the explainablity provided by the traditional Euclidean nearest neighbors, with additional advantages such as dimensionality reduction and understandability-focused visualization.

As future work we plan to further investigate the CMOML optimization method, in order to handle even larger datasets than those used in this work. In particular, image datasets for ordinal regression are the order of the day [61, 62], and dealing with them at the pixel level using Mahalanobis distances is computationally challenging. In this situation it may be interesting to explore the behaviour of the algorithm when it receives the feature maps extracted by a convolutional network [63]. For the purpose of improving the optimization method, we will also explore both the use of optimizers with higher convergence speed and programming paradigms and architectures that maximize the parallel evaluation of the algorithm [64].

Acknowledgements

Our work has been supported by the research project TIN2017-89517-P and by a research scholarship (FPU18/05989), given to the author Juan Luis Suárez by the Spanish Ministry of Science, Innovation and Universities.

Conflict of interest

The authors declare that they have no conflict of interest.

References

[1] D Guijo-Rubio, C Casanova-Mateo, J Sanz-Justo, PA Gutiérrez, S Cornejo-Bueno, C Hervás, and S Salcedo-Sanz. Ordinal regression algorithms for the analysis of convective situations over madrid-barajas airport. *Atmospheric Research*, 236:104798, 2020.

[2] Pavlína Kuráňová. Modelling the results of the phadiatop test using the logistic and ordinal regression. In *Applications of Computational Intelligence in Biomedical Technology*, pages 103–118. Springer, 2016.

[3] Javier Sánchez-Monedero, María Pérez-Ortiz, Aurora Saez, Pedro Antonio Gutiérrez, and César Hervás-Martínez. Partial order label decomposition approaches for melanoma diagnosis. *Applied Soft Computing*, 64:341–355, 2018.

[4] Christopher Beckham and Christopher Pal. Unimodal probability distributions for deep ordinal classification. In *Proceedings of the 34th International Conference on Machine Learning*, pages 411–419, 2017.

[5] Paul-Christian Bürkner and Matti Vuorre. Ordinal regression models in psychology: A tutorial. *Advances in Methods and Practices in Psychological Science*, 2(1):77–101, 2019.

[6] Subhajit Chakraborty and E Mitchell Church. Social media hospital ratings and hcahps survey scores. *Journal of Health Organization and Management*, 2020.

[7] Kostiantyn Antoniuk, Vojtěch Franc, and Václav Hlaváč. V-shaped interval insensitive loss for ordinal classification. *Machine Learning*, 103(2):261–283, 2016.

[8] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018.

[9] Yanzhu Liu, Adams Wai-Kin Kong, and Chi Keong Goh. Deep ordinal regression based on data relationship for small datasets. In *Proceedings of the 26th International Joint Conferences on Artificial Intelligence*, pages 2372–2378, 2017.

[10] Pedro Antonio Gutierrez, Maria Perez-Ortiz, Javier Sanchez-Monedero, Francisco Fernandez-Navarro, and Cesar Hervas-Martinez. Ordinal regression methods: survey and experimental study. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):127–146, 2016.

[11] Eibe Frank and Mark Hall. A simple approach to ordinal classification. In *European Conference on Machine Learning*, pages 145–156. Springer, 2001.

[12] Jianlin Cheng, Zheng Wang, and Gianluca Pollastri. A neural network approach to ordinal regression. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1279–1284. IEEE, 2008.

[13] Wei Chu and S Sathiya Keerthi. Support vector ordinal regression. *Neural computation*, 19(3):792–815, 2007.

[14] Jaime S. Cardoso and Joaquim F. Pinto da Costa. Learning to classify ordinal data: The data replication method. *Journal of Machine Learning Research*, 8(50):1393–1429, 2007.

[15] Hsuan-Tien Lin and Ling Li. Combining ordinal preferences by boosting. In *Proceedings ECML/PKDD 2009 Workshop on Preference Learning*, pages 69–83, 2009.

[16] Yong Shi, Peijia Li, Hao Yuan, Jianyu Miao, and Lingfeng Niu. Fast kernel extreme learning machine for ordinal regression. *Knowledge-Based Systems*, 177:44–54, 2019.

[17] Dan Halbersberg, Maydan Wienreb, and Boaz Lerner. Joint maximization of accuracy and information for learning the structure of a bayesian network classifier. *Machine Learning*, 109:1039–1099, 2020.

[18] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[19] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

[20] Vaishak Belle and Ioannis Papantonis. Principles and practice of explainable machine learning. *arXiv preprint arXiv:2009.11698*, 2020.

[21] Juan Luis Suárez, Salvador García, and Francisco Herrera. A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges. *Neurocomputing*, 425:300–322, 2021.

[22] Zhongchen Ma and Songcan Chen. Multi-dimensional classification via a metric approach. *Neurocomputing*, 275:1121–1131, 2018.

[23] Weiwei Liu, Donna Xu, Ivor W Tsang, and Wenjie Zhang. Metric learning for multi-output tasks. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):408–422, 2018.

[24] Bo Xiao, Xiaokang Yang, Yi Xu, and Hongyuan Zha. Learning distance metric for regression by semidefinite programming with application to human age estimation. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 451–460. ACM, 2009.

[25] Shereen Fouad and Peter Tiňo. Ordinal-based metric learning for learning using privileged information. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2013.

[26] Bac Nguyen, Carlos Morell, and Bernard De Baets. Distance metric learning for ordinal classification based on triplet constraints. *Knowledge-Based Systems*, 142:17–28, 2018.

[27] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[28] Alan Agresti. *Analysis of ordinal categorical data*, volume 656. John Wiley & Sons, 2010.

[29] Ankur Joshi, Saket Kale, Satish Chandel, and D Kumar Pal. Likert scale: Explored and explained. *Current Journal of Applied Science and Technology*, pages 396–403, 2015.

[30] Jean-Baptiste Lamy, Boomadevi Sekar, Gilles Guezennec, Jacques Bouaud, and Brigitte Séroussi. Explainable artificial intelligence for breast cancer: A visual case-based reasoning approach. *Artificial intelligence in medicine*, 94:42–53, 2019.

[31] Bin Gu, Xiang Geng, Wanli Shi, Yingying Shan, Yufang Huang, Zhijie Wang, and Guansheng Zheng. Solving large-scale support vector ordinal regression with asynchronous parallel coordinate descent algorithms. *Pattern Recognition*, 109(107592), 2020.

[32] Hsuan-Tien Lin and Ling Li. Reduction from cost-sensitive ordinal ranking to weighted binary classification. *Neural Computation*, 24(5):1329–1367, 2012.

[33] Gonen Singer, Roee Anuar, and Irad Ben-Gal. A weighted information-gain measure for ordinal classification trees. *Expert Systems with Applications*, 152(113375), 2020.

[34] Rizal Fathony, Mohammad Ali Bashiri, and Brian Ziebart. Adversarial surrogate losses for ordinal regression. In *Advances in Neural Information Processing Systems*, pages 563–573, 2017.

[35] Arthur Mensch, Mathieu Blondel, and Gabriel Peyré. Geometric losses for distributional learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 4516–4525, 2019.

[36] Víctor Manuel Vargas, Pedro Antonio Gutiérrez, and César Hervás-Martínez. Cumulative link models for deep ordinal classification. *Neurocomputing*, 401:48–58, 2020.

[37] Sahibsingh A Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, (4):325–327, 1976.

[38] Tomasa Calvo and Gleb Beliakov. Aggregation functions based on penalties. *Fuzzy sets and Systems*, 161(10):1420–1436, 2010.

[39] Mengzi Tang, Raúl Pérez-Fernández, and Bernard De Baets. Fusing absolute and relative information for augmenting the method of nearest neighbors for ordinal classification. *Information Fusion*, 56:128–140, 2020.

[40] I.T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, 2002.

[41] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2013.

[42] Salvador García, Julián Luengo, and Francisco Herrera. *Data preprocessing in data mining*. Springer, 2015.

[43] Lorenzo Torresani and Kuang-chih Lee. Large margin component analysis. In *Advances in neural information processing systems*, pages 1385–1392, 2007.

[44] Bac Nguyen, Carlos Morell, and Bernard De Baets. Supervised distance metric learning through maximization of the jeffrey divergence. *Pattern Recognition*, 64:215–225, 2017.

[45] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[46] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE transactions on evolutionary computation*, 15(1):4–31, 2010.

[47] Marek Gagolewski. Ordinal regression benchmark data. `https://www.gagolewski.com/resources/data/ordinal-regression/`, 2020. Accessed: 2020-09-10 (YYYY-MM-DD).

[48] Isaac Triguero, Sergio González, Jose M Moyano, Salvador García, Jesús Alcalá-Fdez, Julián Luengo, Alberto Fernández, Maria José del Jesús, Luciano Sánchez, and Francisco Herrera. Keel 3.0: an open source software for multi-stage analysis in data mining. *International Journal of Computational Intelligence Systems*, 10(1):1238–1249, 2017.

[49] Manuel Cruz-Ramírez, César Hervás-Martínez, Javier Sánchez-Monedero, and Pedro Antonio Gutiérrez. Metrics to guide a multi-objective evolutionary algorithm for ordinal classification. *Neurocomputing*, 135:21–31, 2014.

[50] Mithat Gönen and Glenn Heller. Concordance probability and discriminatory power in proportional hazards regression. *Biometrika*, 92(4):965–970, 2005.

[51] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[52] Juan Luis Suárez, Salvador García, and Francisco Herrera. pydml: A python library for distance metric learning. *Journal of Machine Learning Research*, 21(96):1–7, 2020.

[53] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[54] Jeong-Soo Park. Optimal latin-hypercube designs for computer experiments. *Journal of statistical planning and inference*, 39(1):95–111, 1994.

[55] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research*, 18(1):2653–2688, 2017.

[56] Alessio Benavoli, Giorgio Corani, Francesca Mangili, Marco Zaffalon, and Fabrizio Ruggeri. A bayesian wilcoxon signed-rank test based on the dirichlet process. In *International conference on machine learning*, pages 1026–1034, 2014.

[57] Jacinto Carrasco, Salvador García, María del Mar Rueda, and Francisco Herrera. rnpbst: An r package covering non-parametric and bayesian statistical tests. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 281–292. Springer, 2017.

[58] Javier Sánchez-Monedero, Pedro Antonio Gutiérrez, and María Pérez-Ortiz. Orca: A matlab/octave toolbox for ordinal regression. *Journal of Machine Learning Research*, 20(125):1–5, 2019.

[59] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 2018.

[60] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

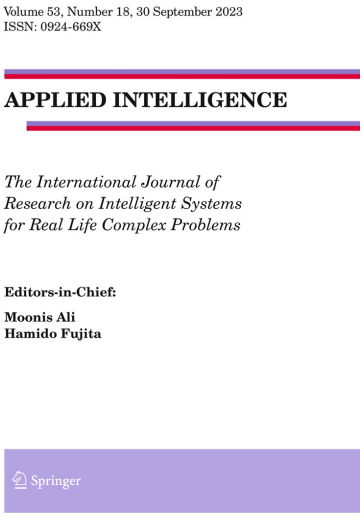
[61] Wanhua Li, Jiwen Lu, Jianjiang Feng, Chunjing Xu, Jie Zhou, and Qi Tian. Bridgenet: A continuity-aware probabilistic network for age estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1145–1154, 2019.

[62] Raul Diaz and Amit Marathe. Soft labels for ordinal regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4738–4747, 2019.

[63] Renqiang Min, David A Stanley, Zineng Yuan, Anthony Bonner, and Zhaolei Zhang. A deep non-linear feature mapping for large-margin knn classification. In *2009 Ninth IEEE International Conference on Data Mining*, pages 357–366. IEEE, 2009.

[64] Mario A Muñoz, Yuan Sun, Michael Kirley, and Saman K Halgamuge. Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. *Information Sciences*, 317:224–245, 2015.

4 Metric learning for monotonic classification: turning the space up to the limits of monotonicity



- **Journal:** Applied Intelligence (APIN)
- **JCR Impact Factor:** 5.3
- **Rank:** 48/145
- **Quartile:** Q2
- **Category:** Computer Science, Artificial Intelligence
- **Status:** Submitted (R1)

Ref.: Suárez, J. L., González-Almagro, G., García, S. & Herrera, F. (2023). Metric learning for monotonic classification: turning the space up to the limits of monotonicity.

A preliminary version of this paper was published at the 2022 35th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA/AIE) with title: A preliminary approach for using metric learning in monotonic classification (DOI: https://doi.org/10.1007/978-3-031-08530-7_65). This conference was held in Kitakyushu, Japan.

METRIC LEARNING FOR MONOTONIC CLASSIFICATION: TURNING THE SPACE UP TO THE LIMITS OF MONOTONICITY

Juan Luis Suárez [*,a,b] **Germán González Almagro** [a,b] **Salvador García** [a,b]

Francisco Herrera [a,b]

[a] *Department of Computer Science and Artificial Intelligence (DECSAI), University of Granada, Granada, Spain*
[b] *Andalusian Institute of Data Science and Computational Intelligence (DaSCI)*

ABSTRACT

This paper presents, for the first time, a distance metric learning algorithm for monotonic classification. Monotonic datasets arise in many real-world applications, where there exist order relations in the input and output variables, and the outputs corresponding to ordered pairs of inputs are also expected to be ordered. Monotonic classification can be addressed through several distance-based classifiers that are able to respect the monotonicity constraints of the data. The performance of distance-based classifiers can be improved with the use of distance metric learning algorithms, which are able to find the distances that best represent the similarities among each pair of data samples. However, learning a distance for monotonic data has an additional drawback: the learned distance may negatively impact the monotonic constraints of the data. In our work, we propose a new model for learning distances that does not corrupt these constraints. This methodology will also be useful in identifying and discarding non-monotonic pairs of samples that may be present in the data due to noise. The experimental analysis conducted, supported by a Bayesian statistical testing, demonstrates that the distances obtained by the proposed method can enhance the performance of several distance-based classifiers in monotonic problems.

Keywords Distance Metric Learning · Monotonic Classification · Nearest Neighbors · Triplet Loss · M-Matrix

* Corresponding Author (`jlsuarezdiaz@decsai.ugr.es`)
Email addresses: `germangalmagro@ugr.es` (Germán González) `salvagl@decsai.ugr.es` (Salvador García), `herrera@decsai.ugr.es` (Francisco Herrera)

1 Introduction

Monotonic constraints [1] are common in real-world prediction problems where the variables to be predicted are ordinal and their order depends on the input data. For example, when predicting house prices, it is expected that—all other things being equal—a bigger house in the same area will have a higher price. Similarly, in predicting students' final grades, students with consistently higher grades during a course are also expected to have a higher final grade. These problems are known as monotonic classification problems [2], and are relevant in fields such as credit risk modeling [3] and lecturer evaluation [4]. Monotonic problems are prevalent in many heavily-regulated industries, and incorporating reasonable expectations about consistent application of selection constraints into automated decision-making systems [5] is crucial [6, 7].

When dealing with these problems, accuracy is not the sole factor to consider. It is equally crucial that the predictions closely follow the monotonic constraints present in the data. Furthermore, the cost of an incorrect prediction should increase as the prediction deviates further away from its actual value. Consequently, there is a need for classifiers that can handle these constraints and factor them in while making predictions.

Ordinal regression methods [8] are commonly used in classification problems where the labels possess an inherent ordering. These methods, which continue to be widely popular today [9, 10, 11], can be particularly useful for monotonic data as the labels in such data also have an inherent ordering. However, ordinal regression methods are not designed to handle monotonic constraints unless they are tailored to that purpose. Despite the significance of monotonicity in several real-world applications, only a few ordinal regression methods specifically address this property. Therefore, further research is necessary to develop more effective and efficient methods for monotonic classification. In recent years, there has been a growing effort to develop new methods for monotonic classification by adapting prominent algorithms from nominal classification, such as decision trees [12] or random forest [13], while also striving to enhance the explainability of the models [14].

Similarity-based learning methods have been successful in monotonic classification problems [2]. This type of learning is inspired by the human ability to recognize objects by their resemblance to other previously seen objects. This idea can be extended to fulfill monotonicity constraints by restricting similar objects or instances to those that comply with these constraints. The well-known nearest neighbors rule for classification [15] has been extended following this idea, so that the nearest neighbors are filtered in order to meet the monotonic constraints [16]. Recently, a new proposal restated the previous idea using a fuzzy approach [17] aiming to gain robustness against possible noise in the monotonicity constraints.

All of the above algorithms require a distance metric to function, and standard distances such as the Euclidean distance have become the go-to choice. However, using a distance metric that is better suited to the data can improve classifier performance. Distance metric learning [18] accomplishes this task and has been successful in several advanced learning problems, such as multi-output learning [19] or multi-dimensional classification [20], as well as ordinal problems with no monotonic constraints [21, 22]. However, its application

when monotonicity constraints are present adds a significant hurdle. Distance metrics have the ability to transform the space [23] and, while this can reduce the number of instances that may break the monotonicity of the dataset, it is difficult to ensure that no new false monotonic constraints are introduced in the process—which may worsen the quality of the data. Although preprocessing techniques such as feature selection methods [24, 25] are effective in monotonic classification problems, the same cannot be said for preprocessing techniques that have the potential to modify the interdependence among features. Consequently, the application of distance metric learning algorithms becomes challenging, making it hard to enhance distance-based classifiers.

Our research presents a novel distance metric learning algorithm for monotonic classification. This algorithm aims to transform the input space in such a way that no new monotonic constraints are introduced, thus resolving the earlier issue. We accomplish this objective through monotonic matrices and M-matrices [26], which possess unique characteristics for defining distances that are highly advantageous for monotonic datasets. As we proceed further into this paper, we will delve deeper into these properties.

This paper represents an extension of our previous work on distance metric learning for monotonic classification [27]. While our earlier paper focused on the development of the basic algorithm and its initial evaluation, this paper presents a comprehensive analysis of the method that includes an expanded description of the method, a further analysis of the background and a theoretical justification of our approach. Our work also provides an extensive experimental evaluation of the method. Specifically, we have conducted a Bayesian statistical analysis of the results and performed a hyperparameter analysis to explore the impact of different parameter settings on the performance of the algorithm. We consider the most relevant metrics in monotonic classification to measure classification performance and test constraint fulfillment after applying our proposed transformations.

The paper is organized as follows. Section 2 describes the current state of distance metric learning and monotonic classification from a similarity-based learning perspective. Section 3 outlines our proposal of distance metric learning for monotonic classification. Section 4 describes the experiments conducted to evaluate the performance of our algorithm, and the results obtained, including the Bayesian statistical analysis and the hyperparameter discussion. Finally, Section 5 ends with the concluding remarks.

2 Background

In this section we will discuss the main problems we have tackled in this paper: distance metric learning, monotonic classification and how similarity-based methods are employed to address monotonic classification nowadays.

2.1 Distance metric learning

Distance metric learning [18] arose with the purpose of improving similarity-based (or, equivalently, distance-based) learning methods such as the k-nearest neighbors classifier,

or k-NN. For this purpose, distance metric learning aims at learning distances that facilitate the detection of hidden properties in the data that a standard distance, such as the Euclidean distance, would fail to discover. Here, we will define *distance* as any map $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ is a non-empty set, satisfying the following conditions:

1. Coincidence: $d(x, y) = 0 \iff x = y$, for every $x, y \in \mathcal{X}$.
2. Symmetry: $d(x, y) = d(y, x)$, for every $x, y \in \mathcal{X}$.
3. Triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$, for every $x, y, z \in \mathcal{X}$.

We will also consider as distances the so-called *pseudo-distances*, which are those maps that verify (2) and (3), and where $d(x, x) = 0$ instead of (1).

Linear distance metric learning is the most common approach to learning distances between numerical data. It consists in learning Mahalanobis distances, which are parameterized by positive semidefinite matrices. Given a positive semidefinite matrix $M \in \mathcal{M}_d(\mathbb{R})_0^+$, and $x, y \in \mathbb{R}^d$, the Mahalanobis distance between x and y defined by M is given as

$$d_M(x, y) = \sqrt{(x - y)^T M (x - y)}.$$

Since every positive semidefinite matrix M can be decomposed as $M = L^T L$ with $L \in \mathcal{M}_d(\mathbb{R})$ it follows that

$$d_M(x, y)^2 = (x - y)^T M (x - y) = (x - y)^T L^T L (x - y) = (L(x - y))^T (L(x - y)) = \|L(x - y)\|_2^2.$$

Therefore, learning a Mahalanobis distance is equivalent to learning a linear map L and then measuring the Euclidean distance after applying that linear map. Thus, the linear distance metric learning approach comes down to learning a positive semidefinite matrix (also called metric matrix) M or a linear map matrix L. Both approaches are equivalent. Learning M usually facilitates convexity during the optimization, while learning L facilitates other tasks such as dimensionality reduction [28].

2.2 Monotonic classification

Monotonic classification [2] arises in certain types of problems of ordinal nature with two particularities: firstly, there are order relations in both the input data (samples) and the output data (labels); secondly, for any given pair of instances, their relative order is also expected to be present in the relative order of their class labels. This happens, for example, when the data represent different measures or evaluations on a particular topic and the label represents a global expert assessment. It is to be expected that, if the measures of one instance are better than the measures of another instance, the global assessment obtained should also be better.

We now formally define what a monotonic dataset is. Let $X = \{x_1, \dots, x_N\} \subset \mathbb{R}^d$ be a numerical dataset. Let $y_1, \dots, y_N \in \{1, \dots, C\}$ be the corresponding labels. The labels can

be ordered using the ordinal relation $\leq$ among the natural numbers, since they take values between 1 and C. For each pair of samples in X, we can also compare their features element-wise. We may not be interested in making all the features comparable, since the monotonic constraints affecting the data may not be present in all the attributes. Thus, let $d_1, \dots, d_m \in \{1, \dots, d\}$ be the indices of all the features that have monotonicity constraints. These constraints can be direct or inverse. Without loss of generality, we can assume all the constraints are direct, and otherwise we can just flip the sign of the affected attribute.

Given two pairs of samples $x_i, x_j \in X$, we define an order relation between them as the product order, considering only the features with monotonicity constraints, i. e.,

$$x_i \leq x_j \iff x_{il} \leq x_{jl}, \text{ for every } l \in \{d_1, \dots, d_m\}.$$

Observe that this order is a partial order, that is, there may be samples x_i, x_j such that $x_i \not\leq x_j$ and $x_i \not\geq x_j$ simultaneously. The dataset $D = \{(x_1, y_1), \dots, (x_N, y_N)\}$ will be *monotonic* if, for every $x_i, x_j \in X$, then

$$x_i \leq x_j \iff y_i \leq y_j.$$

In other words, the dataset D is monotonic if, and only if, for every comparable pair of samples, it is simultaneously true that: (i) all the attributes with monotonic constraints of the first instance are lower or equal than the attributes from the second instance; and (ii) the label of the first instance is lower or equal than the label of the second instance.

It is important to remark that, in real scenarios, due to the subjective nature of the labeling process or to measurement errors, some datasets may not be fully monotonic and there may be several pairs of instances for which monotonicity is broken. In any case, the goal of *monotonic classification* is to provide algorithms that, when predicting new labels, are able to respect the monotonicity constraints of the datasets, and that are also robust against monotonicity clashes that may arise when the dataset is not fully monotonic.

2.3 Monotonic classification and similarity-based learning methods

Similarity-based learning can be seen as closely related to ordinal classification problems. Typically, it is to be expected that if two samples are close their labels will also be close, and the farther apart the samples are the more different their labels will be as well. The k-NN classifier can be easily adapted to this setup. A common approach to handle ordinal labels with this classifier is to modify the aggregation vote function for the nearest neighbors, using, for example, the median of the labels instead of the mode. This can also be extended to handle situations where additional information beyond the labeled data is available [29]. In general, similarity-based algorithms are beneficial in other problems related to ordinal data, including ranking [30].

When our data also have monotonic constraints, additional caution is necessary, since we want the values predicted by the classifier to satisfy these constraints as far as possible. An immediate extension of the nearest neighbors classifier to monotonic classification problems is the *monotonic k-nearest neighbors* classifier (Mon-k-NN), which takes into account only

the nearest neighbors whose labels lie on an interval that does not violate the monotonicity constraints [16]. Given a sample $x_0 \in \mathbb{R}^d$ we can consider the interval $[y_{\min}, y_{\max}]$, where

$$y_{\min} = \max\{y \in \{1, \dots, C\} : (x, y) \in D \text{ and } x \leq x_0\}$$
$$y_{\max} = \min\{y \in \{1, \dots, C\} : (x, y) \in D \text{ and } x_0 \leq x\}$$

Two variants of Mon-k-NN can be considered. The *in-range* (IR) variant considers the k-nearest neighbors to x_0 with labels in the interval $[y_{\min}, y_{\max}]$, while the *out-range* (OR) variant considers the k-nearest neighbors in D and then only those neighbors with labels in $[y_{\min}, y_{\max}]$ are factored in for the vote (if no neighbors have labels in this range, then a random label in the interval will be chosen). Observe that this algorithm will not work properly if the dataset is not fully monotonic. In such a case, $y_{\min}$ may be greater than $y_{\max}$. Therefore, it is necessary to apply a relabeling process that makes the dataset fully monotonic while disturbing it as little as possible. A relabeling method that is applied on the complement of the maximum independent set of the monotonicity violation graph is proposed in [16].

A more recent proposal [17] relies on the fuzzy k-NN [31] in an effort to gain robustness against monotonicity constraints. The *monotonic fuzzy k-nearest neighbors* classifier (Mon-F-k-NN) first uses the in-range monotonic k-nearest neighbors to compute class membership probabilities for each sample in the training set. For each $x_i \in \mathcal{X}$ and $c \in \{1, \dots, C\}$, the probability $u(x_i, c)$ that the class of x_i will be c is defined as

$$u(x_i, c) = \begin{cases} RCr + (nn_c/k)(1 - RCr), & \text{if } y_i = c \\ (nn_c/k)(1 - RCr), \end{cases}$$

where nn_c is the number of nearest neighbors of the class c and RCr is a *real class relevance* estimation, between 0 and 1 (typically established as 0.5). From these memberships, each sample is reassigned to a class whose probability is a median value within the list of membership probabilities for the sample. This class reassignment enhances the monotonicity of the dataset. Finally, at the prediction stage, given the sample x, its k monotonic nearest neighbors $x_{i_1}, \dots, x_{i_k}$ are found and used to compute the membership probabilities of x as

$$u(x, l) = \frac{\sum_{j=1}^{k} u(x_{i_j}, l) \frac{pOR_j}{\|x - x_{i_j}\|^{m-1}}}{\sum_{j=1}^{k} \frac{pOR_j}{\|x - x_{i_j}\|^{m-1}}}.$$

Again, both in-range and out-range variants are available at the prediction stage. The out-range variant considers all the neighbors, even if their labels are not in $[y_{\min}, y_{\max}]$. If this is the case, then pOR_j is set to a previously fixed *out-range penalty* that decides how much weight these neighbors will have in the computation of the membership. In any other case, $pOR_j = 1$. The parameter m determines the influence of the distances of the neighbors. Lastly, the final class of x is taken again using the class associated with the median membership probability in $u(x, \cdot)$.

2.4 Monotonic classification and distance metric learning

Learning a Mahalanobis distance for a monotonic classification problem has several difficulties to overcome. If we try to learn the distance using a metric matrix, the distance is modified while the dataset is not, thus its monotonicity remains unchanged. This is not entirely positive, since the potential of distance metric learning gets squandered and, therefore, also the possibility of reducing the non-monotonicity of the dataset if it exists. However, if we learn the distance using a linear transformation, there is no guarantee that new false monotonic constraints are added. This may happen if we pick a distance defined by a generic $L \in \mathcal{M}_d(\mathbb{R})$. Consider, for example, the extreme case of a matrix L defining a 90-degree rotation in $\mathbb{R}^2$. If such a matrix transforms the dataset, all the monotonic constraints of the original dataset are lost and, furthermore, all those pairs of instances that were not comparable become false monotonic constraints with this rotation.

These drawbacks have so far prevented the development of distance metric learning algorithms for monotonic classification. To the best of our knowledge, there are currently no proposals in this area.

3 Algorithm Description

In this section we will describe our distance metric learning proposal for monotonic classification. First, we will introduce the concepts needed to apply the algorithm. Then, we will describe the algorithm and, finally, we will show its optimization procedure. We named this approach *Large Margin Monotonic Metric Learning* (LM^3L).

3.1 Preliminary definitions

We will focus on the case where all the features in the dataset are subject to direct monotonicity constraints, so that the order relationship in the dataset coincides with the product order in $\mathbb{R}^d$. It's important to note that if there are inverse monotonicity constraints present, we can simply invert the sign of the corresponding features and apply the algorithm to the resultant dataset. Additionally, we will discuss the situation involving non-monotonic features at the end of the section.

As mentioned above, one of the problems of learning a distance by means of a linear transformation is that this transformation disturbs the monotonic constraints and, therefore, some new constraints that are not necessarily true could be added. However, this can be avoided by restricting ourselves to the appropriate subset of matrices, such as the one defined below.

Definition. *A linear transformation or square matrix $L \in \mathcal{M}_d(\mathbb{R})$ is said to be* monotone *[26] if for any real vector $x \in \mathbb{R}^d$, we have that*

$$Lx \geq 0 \implies x \geq 0,$$

where $0 \in \mathbb{R}^d$ is the vector with zeros in all its entries and $\geq$ is the product order in $\mathbb{R}^d$.

Observe that, if L is monotone, if we have two samples $x_i, x_j \in \mathcal{X}$ so that $Lx_i \geq Lx_j$, then $L(x_i - x_j) \geq 0$ and therefore $x_i \geq x_j$. This means that any pair of samples that meets a monotonicity constraint after applying L was already meeting the constraint before applying the transformation. So, when L is monotone, no new monotonic constraints can be added after the dataset is transformed. However, this property is not reciprocal: if $x_i \geq x_j$, it does not necessarily follow that $Lx_i \geq Lx_j$. Consequently, some monotonic constraints may be lost in this transformation. This will allow the algorithms that use this type of matrices to select the constraints that may be more relevant in the dataset without ever adding new incorrect monotonicity constraints after applying the transformation.

Monotone matrices are tough to use in optimization settings, since they cannot be adequately parameterized for this purpose. When L is invertible and monotonic, L is the inverse of a positive matrix (that is, a matrix with all its entries greater than or equal to zero). This may facilitate its parameterization, but the computation of the inverse matrix would make the optimization procedure very expensive. However, there is a subset of monotone matrices with much more suitable properties for use in differential optimization. We describe them below.

Definition. *A linear transformation or square matrix $L \in \mathcal{M}_d(\mathbb{R})$ is an* M-Matrix *[26] if it can be expressed as $L = sI - B$, where I is the identity matrix of dimension d, $B \in \mathcal{M}_d(\mathbb{R})$ is a positive matrix, and $s \in \mathbb{R}$ verifies that $s \geq \rho(B)$, where $\rho(B)$ is the spectral radius of the matrix B.*

M-matrices are monotone [26] and, since they depend on the real value s and the positive matrix B, they can be easily and efficiently used to optimize a differentiable objective function.

3.2 Objective function and optimization

After establishing the linear applications that enable us to regulate the monotonicity of the dataset, the next step is to define the function to be optimized. Since the linear application already controls the monotonicity implicitly, the focus of the objective function will be on assessing a goodness-of-classification metric. This metric should consider the ordinal nature of the dataset, in that the prediction penalty should increase as the actual label moves farther away from the predicted label.

Drawing inspiration from the large margin proposals for distance metric learning in other classification tasks [22, 32], we present a triplet-based objective function. For each anchor sample x_i in the dataset, we consider a positive sample x_j and a negative sample x_l such that $y_i \leq y_j < y_l$ or $y_i \geq y_j > y_l$. The aim is to minimize the distance from x_i to x_j while simultaneously maximizing the distance from x_i to x_l. The objective function and the associated constrained optimization problem are defined as follows:

$$\begin{aligned}\min_{L\in\mathcal{M}_d(\mathbb{R})} f(L) = &\sum_{x_i\in\mathcal{X}} \sum_{\substack{x_j,x_l\in\mathcal{U}(x_i)\\ y_i\le y_j<y_l \\ \text{or} \\ y_i\ge y_j>y_l}} \left[\|L(x_i-x_j)\|^2 - \|L(x_i-x_l)\|^2 + \lambda\right]_+ \\ \text{s. t. }: &L = sI - B \\ &B_{ij}\ge 0, (i,j=1,\dots,d) \\ &s\ge\rho(B).\end{aligned} \tag{1}$$

In the aforementioned optimization problem, the notation $[z]_+ = \max\{z, 0\}$ is used, where λ denotes a margin constant. The aim is to ensure that the distance from the negative sample to the anchor sample is not smaller than the distance from the positive sample to the anchor sample plus the margin constant. Moreover, for each $x_i \in \mathcal{X}$, $\mathcal{U}(x_i)$ represents a neighborhood that includes the K nearest neighbors to x_i for the Euclidean distance. This neighborhood is computed prior to the optimization process and serves to filter the instances that are initially farther away, giving a local character to the method and reducing the computational cost. This draws inspiration from the metric learning method for ordinal regression proposed in [22]. The parameter K represents a hyperparameter that can be adjusted to enhance algorithm performance. It is suggested to set K to a sufficiently large value to ensure representative neighborhoods. This is further discussed in Section 4.5. The choice of the Euclidean distance stems from its suitability as an a priori distance measure before the algorithm learns from the data [32, 33, 34, 35]. However, alternative precomputed distance measures can also be considered.

The constraints specified in the optimization problem of Eq. 1 guarantee that no additional monotonic constraints are introduced when the dataset is transformed. On the other hand, the objective function aims to bring data from nearby classes closer while pushing data from distant classes farther apart. By minimizing Eq. 1, the transformed dataset that we obtain has optimal ordinality and monotonicity properties, which can then be learned by a similarity-based classifier.

To optimize Eq. 1, we propose a stochastic projected gradient descent method. Since L is fully parameterized by s and B, the optimization problem can be rewritten as

$$\begin{aligned}\min_{\substack{s\in\mathbb{R},B\in\mathcal{M}_d(\mathbb{R})\\ s\ge\rho(B)\\ B_{ij}\ge 0\ \forall i,j}} f(L) = \sum_{x_i\in\mathcal{X}} \sum_{\substack{x_j,x_l\in\mathcal{U}(x_i)\\ y_i\le y_j<y_l \\ \text{or} \\ y_i\ge y_j>y_l}} &\left[\|(sI-B)(x_i-x_j)\|^2\right. \\ &\left. - \|(sI-B)(x_i-x_l)\|^2 + \lambda\right]_+ .\end{aligned} \tag{2}$$

At each gradient step we can update the pair (s, B) using the partial derivatives. We know that [36]

$$\frac{\partial f}{\partial s}(s,B) = \sum_{x_i \in \mathcal{X}} \sum_{x_j, x_l \in \mathcal{A}_L(x_i)} d_{ij}^T [2sI - (B + B^T)] d_{ij} - d_{il}^T [2sI - (B + B^T)] d_{il}, \tag{3}$$

$$\frac{\partial f}{\partial B}(s,B) = \sum_{x_i \in \mathcal{X}} \sum_{x_j, x_l \in \mathcal{A}_L(x_i)} 2(B - sI)(O_{ij} - O_{il}), \tag{4}$$

where $d_{ij} = x_i - x_j$, $O_{ij} = d_{ij} d_{ij}^T$, and $\mathcal{A}_L(x_i)$ is the set of active (positive, negative) 2-tuples associated with the anchor sample x_i and $L = sI - B$, that is:

$$\begin{aligned}\mathcal{A}_L(x_i) = &\{(x_j, x_l) : x_j, x_l \in \mathcal{U}(x_i), [(y_i \leq y_j < y_l) \text{ or } (y_i \geq y_j > y_l)] \text{ and} \\ &\|L(x_i - x_j)\|^2 - \|L(x_i - x_l)\|^2 + \lambda > 0\}.\end{aligned}$$

From this, in the stochastic gradient descent process, we choose at each step a random sample $x_i \in \mathcal{X}$ and update s and B with the following rules:

$$s_{new} = s_{old} - \eta \sum_{x_j, x_l \in \mathcal{A}(x_i)} d_{ij}^T [2sI - (B + B^T)] d_{ij} - d_{il}^T [2sI - (B + B^T)] d_{il}, \tag{5}$$

$$B_{new} = B_{old} - \eta \sum_{x_j, x_l \in \mathcal{A}(x_i)} 2(B - sI)(O_{ij} - O_{il}), \tag{6}$$

where η is a pre-established learning rate. Since the above update rules do not ensure that s and B meet the constraints to which they are subject, it is necessary to project them into the constrained set. Therefore, after applying the update rules, we convert the negative entries of B to zero and, if s is smaller than $\rho(B)$, we make it equal to $\rho(B)$:

$$\pi(B) = (\tilde{B}_{ij}), \text{ where } \tilde{B}_{ij} = \max\{B_{ij}, 0\}, \text{ for each } i, j = 1, \dots, d. \tag{7}$$

$$\pi(s) = \max\{s, \rho(B)\}. \tag{8}$$

This concludes the optimization process of LM^3L. In short, at each epoch the samples $x_i \in \mathcal{X}$ are chosen randomly. With each of the samples, s and B are updated using the rules from Eqs. 5 and 6 and then projected into valid values with Eqs. 7 and 8. The process is repeated until a maximum of epochs is reached or the algorithm converges. With the final values of s and B, the obtained distance is retrieved by means of the linear transformation $L = sI - B$.

3.3 Benefits of the Method

Our distance metric learning algorithm for monotonic classification offers several advantages from a theoretical perspective. Firstly, it can find new transformed variables in the

latent space that may better capture the monotonicity of the dataset. By learning a distance metric that is specifically tailored to the problem of monotonic classification, without introducing any new fake monotonic constraints, the algorithm can identify new features that are better suited for capturing the underlying monotonic structure of the data. This can lead to better classification performance and a deeper understanding of the relationships between the variables.

Secondly, since no new monotonic constraints can be added but some of them may be removed, our algorithm can help us to filter the dataset and to discover different ways of interaction in the latent space. By removing constraints that are not relevant, the algorithm can provide insight into the structure of the data and help us to discover new relationships among the variables. This can be particularly useful in cases where the data is high-dimensional or complex, and where traditional methods may struggle to identify meaningful patterns [37].

Finally, the new variables in the latent space can contribute further information on how the variables are related and their impact on the monotonicity of the dataset, which can assist in making interpretable and explainable decisions about the data. By providing a more complete picture of the underlying structure of the data, the algorithm can help us to identify important features and relationships that may not be immediately apparent from the raw data. This can be particularly useful in cases where the data is being used to make critical decisions, such as in finance or healthcare, where interpretability and transparency are essential.

In summary, our distance metric learning algorithm offers several major benefits from a theoretical perspective, including the ability to identify new transformed variables in the latent space, the ability to filter and discover new ways of interaction in the data, and the ability to facilitate interpretable and explainable decisions about the data. These benefits make it a powerful tool for researchers and practitioners working in a variety of fields and applications where monotonicity is a key consideration.

3.4 LM^3L and non-monotonic features

In the previous sections we have assumed that all the features in the dataset are subject to monotonic constraints. However, this assumption may not hold in real-world scenarios. In the context of distance-based classification, both the majority and median-vote k-NN classifiers do not consider the monotonic constraints in any sense. On the other hand, the monotonic and monotonic-fuzzy variants assume the monotonicity across all the features [16, 17]. Since our algorithm is designed to learn a distance that respects the dataset's monotonic constraints, it is essential to address how to handle non-monotonic features and how the later classification stage will be affected by them.

LM^3L can be adapted or combined with other algorithms in order to handle non-monotonic features. One approach is to apply LM^3L locally to the monotonic attributes and then employ another distance metric learning algorithm for standard classification [32, 23] locally to the non-monotonic features. Concatenating the obtained maps, represented as a matrix

containing the two locally learned distance matrices as blocks, will yield a global distance metric for use in the classification stage. This method treats the two types of features separately, thus not capturing interactions between monotonic and non-monotonic features.

An alternative approach that does capture the interactions between monotonic and non-monotonic features is to introduce an unconstrained matrix L_0 to the optimization problem of Eq. 1 for the non-monotonic attributes. This introduces the constraint $L = sI - B + L_0$, where L_0 only contains non-zero rows in the positions corresponding to the non-monotonic features. Consequently, the monotonic constraints remain effective for monotonic features, while L_0 removes limitations on exploring the search space for non-monotonic features.

In applying subsequent distance-based classification methods, since monotonic nearest neighbors approaches assume all features are monotonic, it's advisable to rely on standard majority-vote or median-vote classifiers. These classifiers do not assume monotonicity in the data, as those assumptions are already considered during the distance learning stage.

4 Experiments

In this section we describe the experiments we have developed with our algorithm and the results we have obtained.

4.1 Experimental framework

We have assessed the distance metric learned by LM^3L through various distance-based classifiers. All of them are variations of the nearest neighbors classifier, which include: the original k-NN (majority-vote), the median-vote k-NN, the monotonic k-NN (both the in-range and out-range versions), and the monotonic fuzzy k-NN (both the in-range and out-range versions). The standard k-NN is commonly applied in non-ordinal classification problems, while the median-vote k-NN is the natural adaptation for k-NN in ordinal regression, without taking into consideration any monotonic constraints. The remaining k-NN versions refer to the monotonic nearest neighbors approaches discussed in Section 2.3.

The goal of these experiments is to evaluate whether the distance metric learned by LM^3L can improve the performance of k-NN in two ways: (1) classification accuracy when dealing with new data and (2) adherence to the monotonic constraints of the dataset. To achieve this goal, we will compare various versions of k-NN using both the Euclidean distance and the distance learned by LM^3L.

The experiments were conducted using a fixed number of neighbors $k = 9$ for all the k-NN classifiers. The distances computed using each classifier were evaluated through a stratified 5-fold cross validation, which preserves the original class proportions in each fold. Ten different numerical datasets with monotonic constraints from various sources were used in the experiments [38, 39, 8]. To prepare the data for the experiments, any features with inverse monotonic constraints were sign-switched, and a *min-max* normalization to the interval $[0, 1]$ was applied. It is worth noting that some datasets were not completely monotonic, and contained pairs of samples that violated the monotonicity constraints. The datasets se-

Table 1: Datasets used in the experiments.

Dataset	# Samples	# Features	# Classes	Attribute directions	Non-Monotonic pairs / Comparable pairs (%)
autoMPG8	392	7	5	(-,-,-,-,+,+,+)	0.044 / 36.14
car	1728	6	4	All direct (+)	0.246 / 39.67
ERA	1000	4	9	All direct (+)	3.349 / 16.77
ESL	482	4	7	All direct (+)	0.585 / 59.81
LEV	1000	4	5	All direct (+)	1.330 / 24.08
machineCPU	209	6	5	(-,+,+,+,+,+)	1.196 / 46.24
pima	768	8	2	All direct (+)	0.151 / 6.576
SWD	1000	10	4	All direct (+)	0.949 / 12.62
balance	625	4	3	(-,-,+,+)	0.000 / 25.64
boston-housing	506	13	5	(-,+,-,+,-,+,-,+,-,-,-,+,-)	0.299 / 14.60

lected for the experiments, along with their dimensions and monotonicity properties, are presented in Table 1.

4.2 Metrics and results

To evaluate the classification performance of the distances with each classifier, we have used two metrics: the *mean absolute error* (MAE) [8], which penalizes the classification error according to the distances between the labels, and the *concordance index* (C-INDEX) [40], which measures the ratio between the number of ordered pairs in both true labels and predictions and the number of all comparable pairs.

For the execution of these experiments, the parameters suggested for LM^3L are as follows: a fixed neighborhood size of 50 for the anchor samples, a maximum of 300 optimization epochs, a neighborhood margin λ of 0.1, and an adaptive learning rate η. The adaptive learning rate starts at 10^{-6} and, at each epoch, it is either increased by 1 % if the objective function improves, or halved if it does not, following the adaptive approach in [32]. These parameters were chosen based on the guidelines of the algorithms that inspired this method, as well as a preliminary hyperparameter analysis presented in Section 4.5. The code of LM^3L used for these experiments is available in `pyDML` [41], which is a Python library containing various distance metric learning algorithms.

Table 2 shows the results of the classification performance. This table also includes, for each combination of distance and classifier, its average ranking over all the combinations of distance and classifiers (AVG RANK [ALL]) and its average ranking within the distances that use the same classifier (AVG RANK [IN]).

To evaluate the fulfillment of the monotonic constraints we rely on the *non monotonicity index* (NMI). This metric is a normalized measure of how many samples do not fulfill a monotonic constraint. This can be used to evaluate both the monotonicity of the transformed training dataset after applying LM^3L and the monotonicity of the predicted samples with respect the training dataset. For a training set $\mathcal{X}$ and a labeled point $(x, y) \in \mathbb{R}^d \times \{1, \dots, C\}$

Table 2: MAE and C-INDEX of the distance and classifiers on each dataset.

Dataset	k-NN		Med-k-NN		Mon-k-NN (IR)		Mon-k-NN (OR)		Mon-F-k-NN (IR)		Mon-F-k-NN (OR)	
	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L
C-INDEX												
autoMPG8	0.918457	0.920312	0.920898	**0.925040**	0.920753	**0.925040**	0.919799	**0.925040**	0.919230	0.911326	0.918868	0.912257
car	0.963044	0.972001	0.962430	**0.974398**	0.971015	**0.974398**	0.970817	**0.974398**	0.964425	0.736149	0.942335	0.718687
ERA	0.692304	0.683062	0.696339	0.692771	0.675817	0.662343	0.675817	0.662343	**0.702484**	0.698284	0.700877	0.701683
ESL	0.911469	0.914401	0.917153	0.918642	0.912686	0.901638	0.911558	0.902055	0.917847	0.915939	**0.918671**	0.916644
LEV	0.803299	0.816627	0.804052	0.819050	0.768287	0.736397	0.768287	0.736397	0.818089	0.815131	**0.822355**	0.821664
machineCPU	0.857039	0.854280	0.860909	0.867868	**0.880012**	0.877510	0.873755	0.874879	0.863247	0.869615	0.855029	0.870145
pima	0.705144	0.712881	0.705144	0.712881	0.706292	0.706473	0.709292	0.706473	**0.719280**	0.667439	0.685984	0.665552
SWD	0.742335	0.743421	0.749917	0.752555	0.685411	0.674434	0.689080	0.674434	0.759628	0.757493	**0.761008**	0.754048
balance	0.903139	0.898828	0.933961	0.940843	0.946904	0.948984	0.946882	0.948657	0.948366	**0.951353**	0.944890	0.944437
boston-housing	0.780491	0.848367	0.795450	**0.853051**	0.803741	**0.853051**	0.799271	**0.853051**	0.819694	0.810555	0.773578	0.810751
AVG SCORE	0.827672	0.836418	0.834625	**0.845710**	0.827092	0.826027	0.826456	0.825773	0.843229	0.813328	0.832360	0.811587
AVG RANK [ALL]	8.863636	6.409091	6.954545	**3.772727**	6.818182	6.454545	7.363636	6.636364	4.090909	7.090909	6.363636	7.181818
AVG RANK [IN]	1.666667	**1.333333**	1.833333	**1.166667**	**1.416667**	1.583333	**1.416667**	1.583333	**1.250000**	1.750000	**1.333333**	1.666667
MAE												
autoMPG8	0.342740	0.322085	0.332608	**0.306889**	0.334720	**0.306889**	0.337281	**0.306889**	0.329528	0.359831	0.331758	0.360091
car	0.052658	0.050339	0.054392	0.049193	**0.030085**	0.049193	0.031826	0.049193	0.056113	0.256990	0.090264	0.270272
ERA	1.411089	1.465980	1.300229	1.310030	1.527210	1.556053	1.527210	1.556053	**1.291049**	1.300199	1.293127	1.292118
ESL	0.331710	0.327648	0.321226	**0.317207**	0.352483	0.396309	0.356799	0.396309	0.342489	0.346530	0.335982	0.338046
LEV	0.438115	0.415123	0.421073	0.405062	0.496170	0.544144	0.496170	0.544144	0.398021	0.407012	0.384050	0.389021
machineCPU	0.613086	0.612891	0.574255	0.560191	**0.489040**	0.513241	0.508696	0.527438	0.549397	0.527777	0.572187	0.543366
pima	0.247441	0.247432	0.247441	0.247432	0.244826	0.251328	**0.240905**	0.251328	0.244835	0.259138	0.247449	0.260453
SWD	0.479069	0.482130	0.448037	0.450157	0.503911	0.536098	0.499966	0.536098	0.442002	0.447038	**0.440997**	0.450057
balance	0.148918	0.158031	0.147267	0.134311	**0.083351**	0.087984	**0.083351**	0.091236	0.092991	0.092927	0.129795	0.129666
boston-housing	0.588659	0.422635	0.549184	**0.406773**	0.499192	**0.406773**	0.517014	**0.406773**	0.672612	0.473747	0.570193	0.477609
AVG SCORE	0.465349	0.450429	0.439571	**0.418725**	0.456099	0.464801	0.459922	0.466546	0.441904	0.447119	0.439580	0.451070
AVG RANK [ALL]	8.590909	6.590909	6.409091	**4.318182**	6.000000	7.227273	6.454545	7.590909	5.363636	6.636364	5.818182	7.000000
AVG RANK [IN]	1.750000	**1.250000**	1.833333	**1.166667**	**1.166667**	1.833333	**1.166667**	1.833333	**1.250000**	1.750000	**1.333333**	1.666667

we define

$$NClash_{\mathcal{X}}(x) = |\{x_i \in \mathcal{X} : (x_i < x \text{ and } y_i > y) \text{ or } (x_i > x \text{ and } y_i < y)\}|.$$

Then, the NMI of the labeled dataset $\mathcal{X}$ with respect to the labeled dataset $\mathcal{Y}$ is defined as

$$NMI(\mathcal{X}, \mathcal{Y}) = \frac{1}{|\mathcal{X}||\mathcal{Y}| - |\mathcal{X} \cap \mathcal{Y}|} \sum_{x \in \mathcal{Y}} NClash_{\mathcal{X}}(x).$$

We can use the NMI in different ways. If we want to measure the monotonicity of the original training set $\mathcal{X}$, we can use $NMI(\mathcal{X}, \mathcal{X})$. If we want to measure the monotonicity of the training set after being transformed by a linear map $L \in \mathcal{M}_d(\mathbb{R})$, we can use $NMI(L\mathcal{X}, L\mathcal{X})$. Finally, if we want to measure the monotonicity of a set of test samples and their predictions, $\mathcal{X}_t$, with respect to the training set, we can use $NMI(L\mathcal{X}, \mathcal{X}_t)$. Table 3 shows the results regarding the fulfillment of the monotonic constraints. In this table, the metric NMI-TRAIN represents the NMI for the training sets, for the Euclidean distance (that is, with no transformations applied) and for the transformed dataset using the distance learned by LM^3L. The NMI-TEST metric represents each of the NMIs of the training sets for each distance, with respect to the sets of predicted values by each of the classifiers for the test set. We also show the total of comparable pairs in the training set (CP-TRAIN), and between the training and test sets (CP-TEST), for each of the distances.

Finally, we also include in Table 5 the time required for LM^3L to learn the distance within each dataset. It is important to note that these times are independent of the classifier employed, as the distance is learned independently of the classification stage. In addition, there

Table 3: Results of the monotonicity analysis on each dataset. The (C.I.) column title states that the metrics do not depend on the classifier used, only on the distance.

Metric	NMI-TRAIN		CP-TRAIN		NMI-TEST					
Classifier	(C.I.)		(C.I.)		k-NN		Med-k-NN		Mon-k-NN (IR)	
Distance	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L
autoMPG8	0.002558	**0.000000**	**0.401125**	0.000539	0.001534	**0.000000**	0.001390	**0.000000**	0.000834	**0.000000**
car	0.000058	**0.000000**	**0.143672**	0.003914	0.000126	**0.000000**	0.000116	**0.000000**	0.000028	**0.000000**
ERA	0.033560	**0.021917**	**0.167776**	0.075397	0.026396	0.018346	0.025813	0.017834	0.027845	0.020012
ESL	0.009530	**0.003307**	**0.703749**	0.348972	0.006029	0.002217	0.005926	0.002212	0.007266	0.002425
LEV	0.013335	**0.005879**	**0.240869**	0.048001	0.008462	0.003953	0.008401	0.003911	0.010639	0.005150
machineCPU	0.013873	**0.003957**	**0.497229**	0.289424	0.011322	0.002963	0.011395	0.002906	0.006639	0.001817
pima	0.001640	**0.000045**	**0.073187**	0.001316	0.000896	0.000015	0.000896	0.000015	0.000396	0.000002
SWD	0.009505	**0.004307**	**0.126258**	0.008387	0.005609	0.003078	0.005274	0.003100	0.007336	0.003595
balance	**0.000000**	**0.000000**	**0.256326**	0.209876	0.000051	0.000041	0.000038	0.000016	**0.000000**	**0.000000**
boston-housing	0.002775	**0.000000**	**0.149025**	0.000000	0.001934	**0.000000**	0.001731	**0.000000**	0.001443	**0.000000**
AVG SCORE	0.008683	**0.003941**	**0.275922**	0.098583	0.006236	0.003061	0.006098	0.002999	0.006243	0.003300
AVG RANK	1.954545	**1.045455**	**1.000000**	2.000000	10.681818	4.636364	9.772727	4.090909	9.500000	4.318182

Metric	NMI-TEST						CP-TEST	
Classifier	Mon-k-NN (OR)		Mon-F-k-NN (IR)		Mon-F-k-NN (OR)		(C.I.)	
Distance	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L
autoMPG8	0.000762	**0.000000**	0.000747	**0.000000**	0.001657	**0.000000**	**0.400190**	0.000507
car	0.000028	**0.000000**	0.000085	0.000005	0.000222	4.18e-07	**0.143342**	0.003887
ERA	0.027845	0.020012	0.024477	0.017358	0.024352	**0.017335**	**0.167337**	0.075719
ESL	0.007138	0.002404	0.006103	0.001991	0.006096	**0.001914**	**0.703130**	0.351790
LEV	0.010639	0.005150	0.007547	0.003831	0.007555	**0.003815**	**0.240689**	0.047908
machineCPU	0.006346	0.001614	0.008060	**0.001277**	0.011200	0.002779	**0.489219**	0.286028
pima	0.000411	0.000002	0.000231	**0.000000**	0.000744	0.000012	**0.073161**	0.001432
SWD	0.006880	0.003595	0.005230	0.002951	0.005212	**0.002945**	**0.125906**	0.008363
balance	**0.000000**	**0.000000**	**0.000000**	**0.000000**	0.000003	**0.000000**	**0.256840**	0.211008
boston-housing	0.001438	**0.000000**	0.000441	**0.000000**	0.001719	**0.000000**	**0.147432**	0.000000
AVG SCORE	0.006149	0.003278	0.005292	**0.002741**	0.005876	0.002880	**0.274725**	0.098664
AVG RANK	8.954545	4.045455	7.545455	**2.545455**	9.181818	2.727273	**1.000000**	2.000000

Table 4: Non-monotonicity index over the comparable pairs in both train and test datasets.

Metric	NMI on CP [TRAIN]		NMI on CP [TEST]											
Classifier	(C.I.)		k-NN		Med-k-NN		Mon-k-NN (IR)		Mon-k-NN (OR)		Mon-F-k-NN (IR)		Mon-F-k-NN (OR)	
Distance	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L	Euclidean	LM^3L
autoMPG8	0.006381	**0.000000**	0.003834	**0.000000**	0.003473	**0.000000**	0.002084	**0.000000**	0.001904	**0.000000**	0.001870	**0.000000**	0.004162	**0.000000**
car	0.000402	**0.000000**	0.000881	**0.000000**	0.000812	**0.000000**	0.000190	**0.000000**	0.000190	**0.000000**	0.000589	0.001088	0.001530	0.000135
ERA	**0.200070**	0.291700	0.158011	0.243392	0.154546	0.236284	0.166521	0.264883	0.166521	0.264883	0.146549	0.229999	**0.145805**	0.229653
ESL	0.013542	**0.009365**	0.008573	0.006386	0.008426	0.006376	0.010339	0.006775	0.010157	0.006734	0.008683	0.005616	0.008672	**0.005460**
LEV	**0.055372**	0.122519	0.035194	0.082691	0.034919	0.081782	0.044228	0.107630	0.044228	0.107630	**0.031383**	0.080121	0.031412	0.079804
machineCPU	0.027944	**0.013879**	0.023643	0.011360	0.023905	0.011158	0.013765	0.007022	0.013158	0.006294	0.016674	**0.004882**	0.023268	0.010723
pima	**0.022528**	0.044904	0.012296	0.036612	0.012296	0.036612	0.005440	0.000465	0.005628	0.000465	0.003171	**0.000000**	0.010213	0.036147
SWD	0.075348	**0.051356**	0.044585	0.036811	0.041939	0.037076	0.058386	0.042992	0.054865	0.042992	0.041572	0.035292	0.041459	**0.035218**
balance	**0.000000**	**0.000000**	0.000200	0.000257	0.000149	0.000083	**0.000000**	**0.000000**	**0.000000**	**0.000000**	**0.000000**	**0.000000**	0.000012	**0.000000**
boston-housing	**0.018065**	-	**0.013812**	-	**0.012271**	-	**0.010734**	-	**0.010695**	-	**0.003323**	-	**0.012661**	-
AVG SCORE	**0.038232**	0.053438	0.027446	0.041861	0.026692	0.041047	0.028361	0.042976	0.027966	0.042900	**0.023074**	0.035700	0.025394	0.039774
AVG RANK	1.650000	**1.350000**	8.083333	7.636363	7.166666	7.000000	6.916666	6.136363	6.500000	5.863636	4.625000	**4.545454**	6.500000	5.318181

is no comparison with the Euclidean distance, opposed as it was done in Table 2. One might assume that the Euclidean distance requires zero time, but in reality, no distance learning process occurs in that scenario. The provided timings illustrate that the algorithm scales effectively in response to the growing number of samples in the datasets. This can be primarily attributed to the local nature provided by the neighborhood filter during triplet generation.

Table 5: Time (in seconds) required for LM^3L to learn the distance within each dataset.

	TIME
autoMPG8	3053.281696
car	9943.930562
ERA	8204.803218
ESL	2991.658279
LEV	7656.171229
machineCPU	1502.185281
pima	5180.773168
SWD	7346.903710
balance	1818.846962
boston-housing	4512.102694

4.3 Analysis of results

Based on the results presented in Tables 2 and 3, we can make the following observations. Firstly, we can conclude that the performance of the Med-k-NN classifier improves significantly when combined with LM^3L in terms of both MAE and C-INDEX. In fact, the combination of LM^3L and Med-k-NN is the most successful one in the experiments. In contrast, the combination of LM^3L with monotonic classifiers yielded less competitive results compared to non-monotonic classifiers, often performing worse than the Euclidean distance in those particular cases. Thus, we can say that the distance learned by LM^3L is capable of achieving superior classification performance compared to the Euclidean distance, but only when used in combination with the more traditional k-NN and Med-k-NN classifiers. This could be attributed to the fact that the monotonic classifiers already heavily focus on optimizing the constraint aspect, which may render their combination with LM^3L counterproductive. In any case, combining LM^3L with a non-monotonic classifier is not a drawback, since monotonic constraints are already taken into account in the distance learning process itself.

Finally, by looking at the monotonicity results, it becomes evident that the transformation learned by our algorithm significantly reduces the number of non-monotonic pairs of samples after transforming the training set. The observed monotonicity of the predicted samples with respect to the training set confirms that LM^3L is successful in decreasing the number of predictions that violate a monotonic constraint, for all classifiers. This highlights the capability of our method to avoid introducing new incorrect monotonic constraints when transforming the dataset, owing to the utilization of M-matrices in the optimization process. However, it is worth noting that the reduction in NMI comes at the expense of diminishing the number of comparable instances in the dataset, as is apparent in Table 3; the number of comparable pairs is consistently higher in the untransformed dataset. Nevertheless, this reduction can assist in identifying instances that are inaccurately linked in monotonic constraints due to noise or lack of accuracy. The results presented in Table 4 demonstrate the

performance of the NMI metric when considering only the comparable pairs in both the training and test sets. The average NMI values for Euclidean and LM^3L distances are similar, but LM^3L outperforms Euclidean distance in terms of ranking. These findings suggest that, despite the reduction in the number of comparable pairs, the NMI metric normalized by the number of comparable pairs remains competitive when LM^3L is employed.

4.4 Bayesian non-parametric statistical analysis

In order to assess the extent to which the best models obtained outperform the other models, and to compare the distances learned on the same classifier, we have performed a series of Bayesian statistical tests. We have prepared several pairwise Bayesian sign tests [42] to perform these comparisons. The tests take into account the differences between the C-Index and MAE scores obtained by each pair of compared algorithms, assuming that their prior distribution is a Dirichlet process [43], defined by a prior strength $s = 1$ and a prior pseudo-observation $z_0 = 0$. After perceiving the score obtained for each dataset, the tests produce a posterior distribution that gives us the probabilities that either one of the compared algorithms outperforms the other, or that they are practically equivalent. The region of practical equivalence has been established as the region where the score differences are in the interval $[-0.01, 0.01]$. In summary, from the posterior distribution we obtain three probabilities: the probability that the first algorithm outperforms the second, the probability that the second algorithm outperforms the first, and the probability that the two algorithms are practically equivalent. The distribution can be plotted as a ternary simplex plot for a sample of the posterior distribution, where a greater skew of the points towards on of the regions represent a higher probability.

To carry out the Bayesian sign tests we have used the R package `rNPBST` [44]. In Figures 1 and 2 we show all the pairwise comparisons among every ¡combination of distance and classifier. This comparison is displayed as a heatmap, with the lower half showing the posterior probability for the algorithm with the highest likelihood of outperformance against its competitor. The color of the heatmap in this half indicates which algorithm is the winner via an increase in color intensity with higher probability of outperformance. The upper half shows the posterior probabilities that the compared pairs of algorithms are practically equivalent (the rope region probability). Again, the intensity of the color refers to a higher probability, while the two colors indicate how high the rope probability is: whether the algorithms are more likely to perform equivalently or the better algorithm in the lower half clearly wins.

In the comparisons of Figures 1 and 2, we can confirm that Med-k-NN with the distance learned by LM^3L is the algorithm that stands out the most, since, when compared to the other algorithms, its probability of winning always exceeds the probability of the other algorithm winning. Looking at the C-Index, we observe that the rope probabilities are high in general, which indicates that it is also likely that Med-k-NN with LM^3L has an equivalent performance, in terms of the C-Index, to the other algorithms. In any case, the probability that this algorithm will be significantly outperformed by any of the compared algorithms is always lower. As for MAE, we see that the rope probabilities are no longer as high. There-

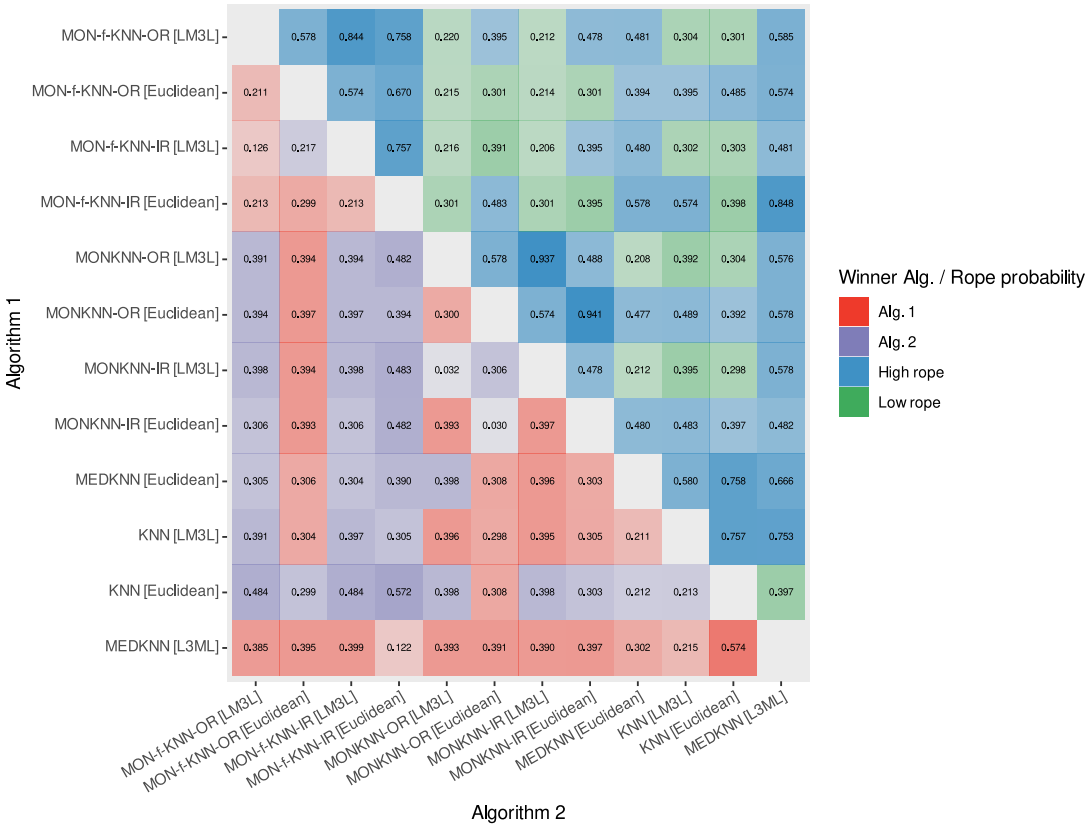


Figure 1: Pairwise Bayesian comparisons of the C-Index scores obtained by the different algorithms.

fore, the probability that $Med-\text{k}-NN$ with LM^3L significantly outperforms any of the compared algorithms, with respect to MAE, is clearly dominant.

The above heatmaps offer a general overview of the Bayesian test results. To have a more specific view we focus now on two main comparisons: one for the best algorithm obtained against the rest of the algorithms, and another one for the Euclidean distances against the distances learned by LM^3L within the same classifier, for each of the classifiers analyzed in this study. For this purpose, we have obtained the ternary simplex plots and the posterior distribution barplots for each of the pairwise comparisons, which are available in Appendix A.

The first comparison with Bayesian tests puts the classification model with Med-k-NN and the distance learned by LM^3L, which is the best performer according to the tables, against the rest of the classifiers and distances. Figures 11-14 show the relevant Bayesian plots.

This comparison confirms what we had already observed in the heatmaps: in all the algorithms there is a clear trend towards the regions associated with Med-k-NN with LM^3L and the rope. In the case of C-Index there is a greater bias towards the rope, while for the MAE it becomes strongly apparent that the distributions are concentrated in the region of Med-k-

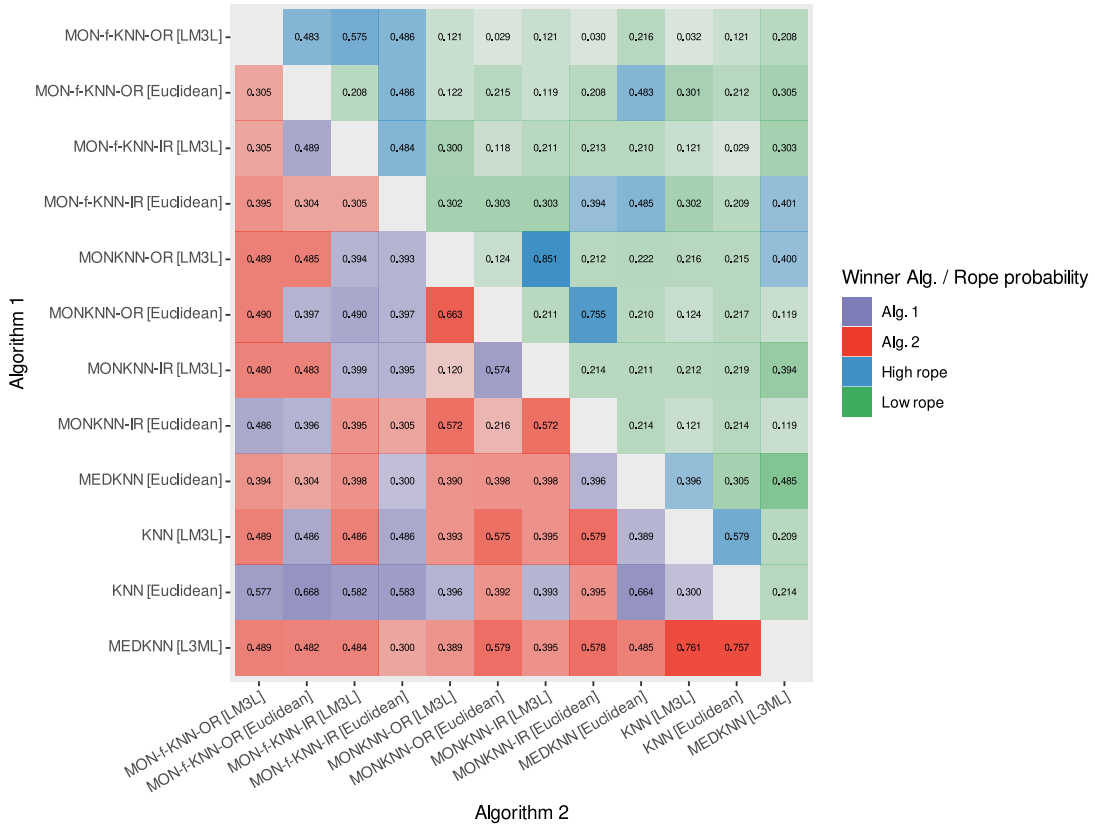


Figure 2: Pairwise Bayesian comparisons of the MAE scores obtained by the different algorithms.

NN with LM^3L, thus showing that this algorithm is most likely significantly outperforming the rest under this metric.

The second analysis with Bayesian tests compares, within the same classifier, the Euclidean distance and the distance learned by LM^3L. Figures 15-16 show the Bayesian plots obtained for this analysis. We can confirm, as already seen in the Tables, that LM^3L is able to outperform the Euclidean distance when using the non-monotonic majority-vote and median-vote nearest neighbor classifiers, although it is not significantly better than the Euclidean distance when comparing within each of the monotonic classifiers. According to the metrics, the MAE shows more bias towards the winner algorithm region, for each case, and the C-Index is more dominated by the rope. In any case, these diagrams show how dominant Med-k-NN with LM^3L is with respect to the Euclidean Med-k-NN. Together with the above comparison, LM^3L is still validated as the best alternative when used in conjunction with the median-vote nearest neighbors.

4.5 Analysis of hyperparameters

In this section, we analyze the hyperparameters of the proposed algorithm on some of the datasets used in the experiments above. The main parameters of LM^3L are:

- The initial learning rate for the gradient optimization η_0.
- The large margin λ in the objective function.
- The neighborhood size K.
- The maximum number of iterations of the gradient optimization.

We evaluate these hyperparameters in the datasets `autoMPG8` and `boston-housing`, which are both inspired by real world monotonic problems. We use LM^3L with the same Med-9-NN classifier used in the experiments.

4.5.1 Influence of the initial learning rate

In what follows, we analyze the impact of the initial learning rate on the above-mentioned datasets. With the rest of the parameters fixed as in the initial experimentation, we vary η_0 according to the set of values $\{10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. The results are shown in Figures 3 and 4.

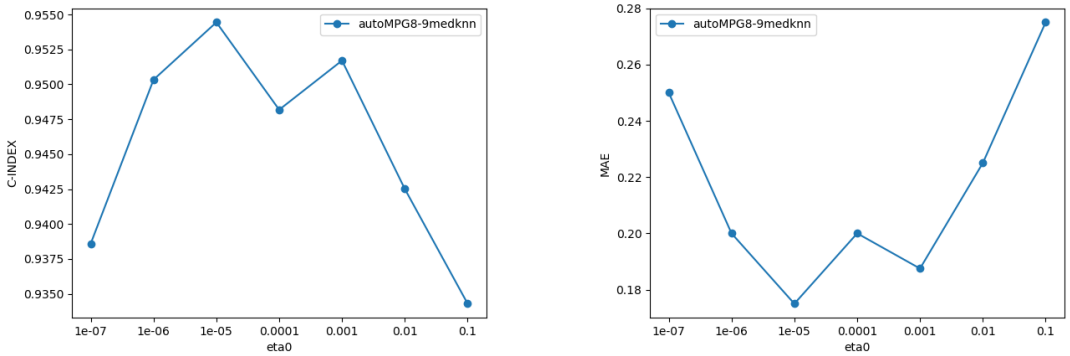


Figure 3: Effect of η_0 on C-Index and MAE in autoMPG8.

In the graphics we can see that, when η_0 ranges from 10^{-6} to 10^{-3}, the adaptive update of the learning rate is enough to lead to competitive results. In contrast, when eta_0 is too low or too high, it negatively influences the optimization process and the final metrics are suboptimal, despite the adaptability of η during the gradient optimization.

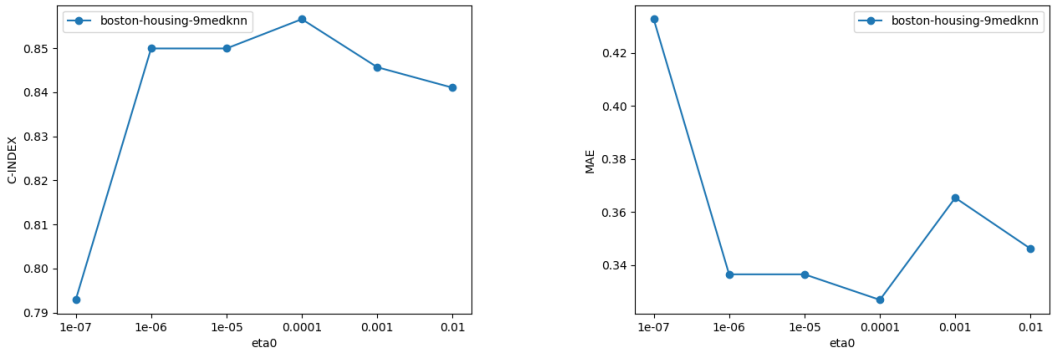


Figure 4: Effect of η_0 on C-Index and MAE in boston-housing.

4.5.2 Influence of the margin

The margin λ determines the degree to which the most distant class is kept away in the triplets that are used during the optimization process. When the margin is low, the three ordered elements in the triplet are closer than when the margin is high. We analyze the impact of λ for the set of values

$$\{0.01, 0.05, 0.1, 0.2, 0.4, 0.6, 0.8, 1.0, 1.5, 2.0, 3.0, 4.0, 5.0, 10.0, 20.0, 30.0, 40.0, 50.0\}$$

with the other parameters fixed, in Figures 5 and 6.

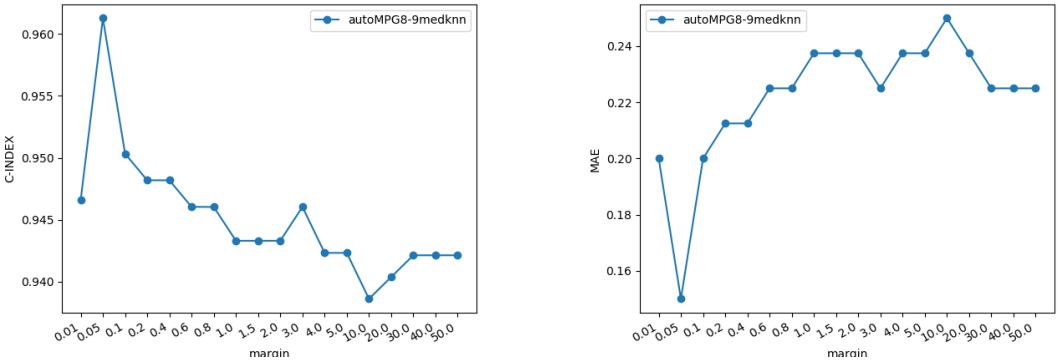


Figure 5: Effect of λ on C-Index and MAE in autoMPG8.

Here, it is readily visible that the highest levels of performance are achieved when the margins are below 1, which tells us that it is interesting to keep the elements of the triplets close together (as long as they are correctly ordered). The amplitude of the optimal margin range seems to be small as well, so it is crucial to specify this margin adequately to achieve optimal performance.

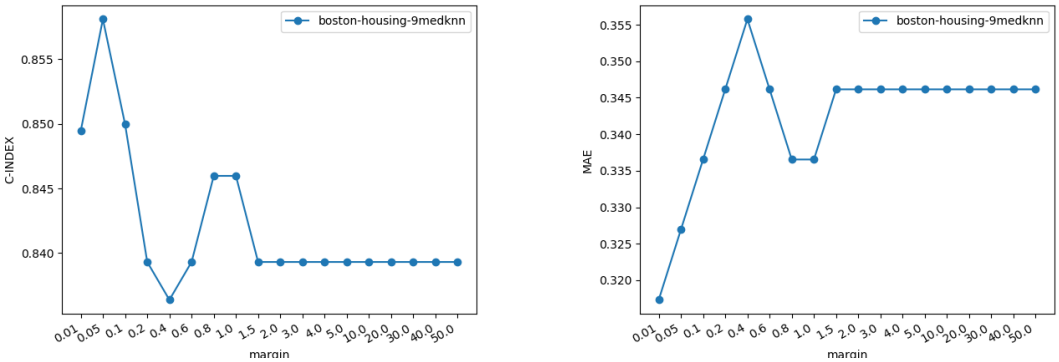


Figure 6: Effect of λ on C-Index and MAE in boston-housing.

4.5.3 Influence of neighborhood size

The neighborhood size K determines how many neighbors are considered to compute the triplets around each anchor sample in the dataset. This parameter gives a local character to the algorithm, as only nearby samples will be considered for each element. If K is low, only a few nearest neighbors will be used to compute the triplets. If K is high, the triplets will take into account most of the dataset. A lower value of K also translates into higher efficiency.

We analyze the impact of the neighborhood size used with the set of values in the range from 10 to 100 with a step of 5. Figures 7 and 8 show the effect of the neighborhood size on the C-Index and MAE.

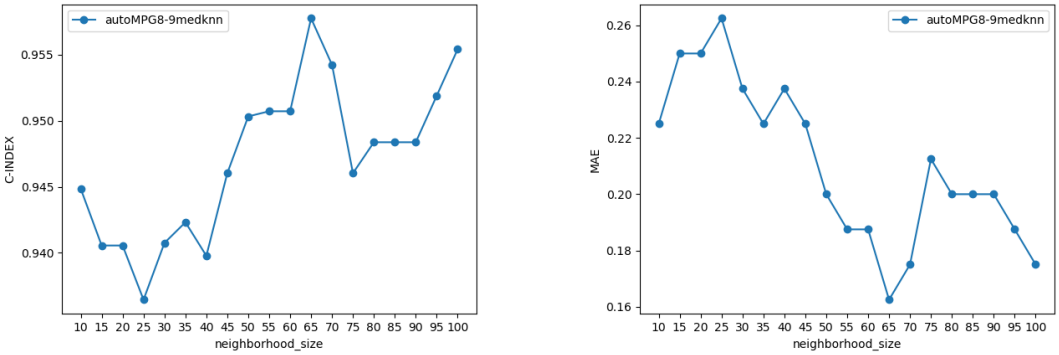


Figure 7: Effect of K on C-Index and MAE in autoMPG8.

In the figures we can observe that a good performance of the algorithm is usually achieved when the neighborhood size is 50 or higher. The optimal value may vary but, in general, a higher quality is obtained when the neighborhood size is in this range.

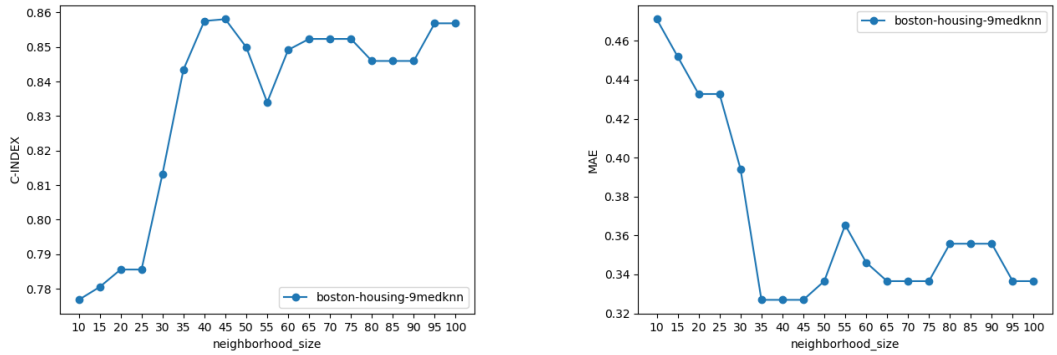


Figure 8: Effect of K on C-Index and MAE in boston-housing.

4.5.4 Influence of the number of iterations.

Lastly, we study how the number of iterations of the gradient optimization affects the convergence of the algorithm. Figures 9 and 10 show the effect of the number of iterations on the C-Index and MAE, in a range from 10 to 500 with a step of 10.

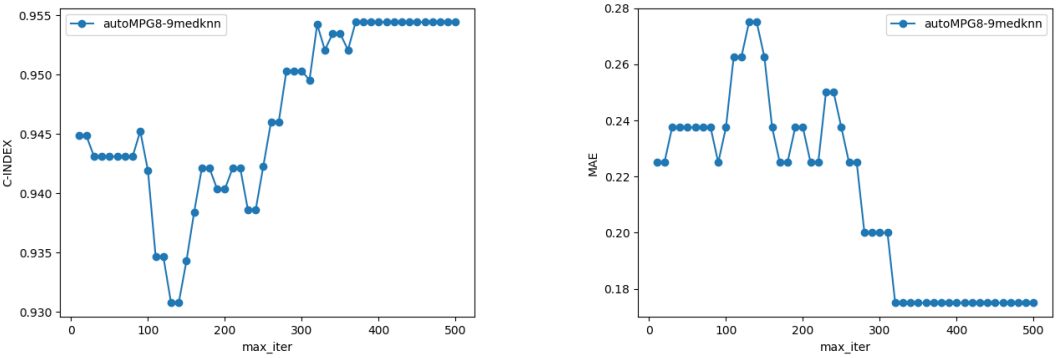


Figure 9: Effect of the number of iterations on C-Index and MAE in autoMPG8.

From the graphics we can conclude that the algorithm seems to converge with a number of iterations around 300, and there does not seem to be overfitting, as a higher number of iterations does not imply a worsening in the values of the metrics in this case.

5 Conclusion

In this paper, we have presented a new distance metric learning algorithm developed specifically for monotonic classification that, for the first time, exploits the potential of linear transformations to reduce the non-monotonicity of the dataset, thanks to the use of M-matrices.

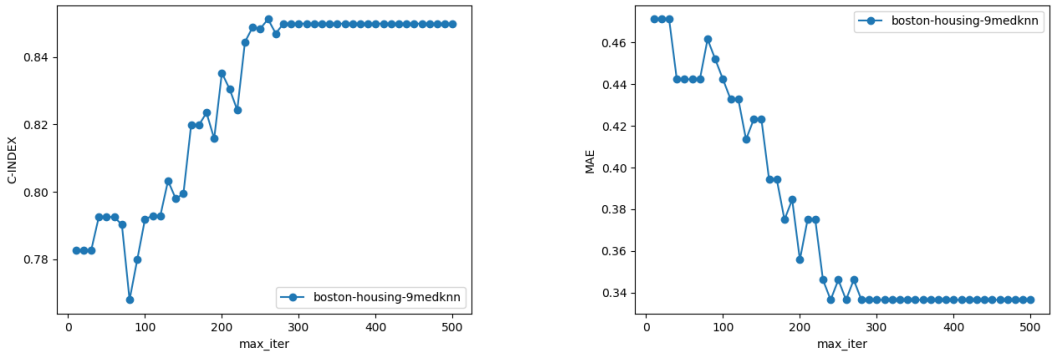


Figure 10: Effect of number of iterations on C-Index and MAE in boston-housing.

In addition, the distances learned allow us to improve the classification performance of the classifiers analyzed.

The results, supported by a Bayesian analysis, have shown that LM^3L combined with the median vote nearest neighbors classifier can outperform even the monotonic distance-based classifiers. In addition, the transformation of the space performed by LM^3L allows the number of non-monotonic pairs in the dataset to be reduced without introducing any false new monotonic constraints. LM^3L is thus presented as an alternative to consider in monotonic classification problems based on distances or similarities.

Acknowledgements

Our work has been supported by the research projects PID2020-119478GB-I00 and A-TIC-434-UGR20, and by a research scholarship (FPU18/05989), given to the author Juan Luis Suárez by the Spanish Ministry of Science, Innovation and Universities.

Authorship contribution statement

Juan Luis Suárez: Conceptualization, Methodology, Software, Investigation, Writing - review & editing, Validation. **Germán González-Almagro:** Conceptualization, Methodology, Writing - review & editing. **Salvador García:** Conceptualization, Methodology, Writing - review & editing, Investigation, Funding acquisition, Supervision. **Francisco Herrera:** Funding acquisition, Project administration, Supervision.

Compliance with ethical standards

The study was conducted in compliance with ethical standards, including obtaining informed consent from all participants and ensuring the confidentiality and anonymity of their data.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethical and informed consent for data used

The data used in this study were publicly available datasets. No ethical approval was required for the use of these datasets. We adhered to best practices for data management, including ensuring the quality and reliability of the data and properly citing the source of the data.

Data availability and access

The datasets used in this work are publicly available in the KEEL [38] repository (`https://sci2s.ugr.es/keel/datasets.php`) and in the ORCA [8] repository (`https://github.com/ayrna/orca`), as a download link (`http://www.uco.es/grupos/ayrna/ucobigfiles/datasets-orreview.zip`). Any of the specific folds used in the experimentation can be requested to the corresponding author on request.

A Bayesian test simplex plots and posterior distributions

This Appendix shows the pairwise Bayesian diagrams for the two comparisons described in Section 4.4. The simplex plots are ternary plots displaying a sample of the posterior distribution. There are three regions that correspond to either algorithm having a performance advantage and to the rope (practical equivalence); each region is centered at one different vertex of the simplex, so that a greater tendency of the points towards a region represents a greater probability for that option. The posterior distributions are also shown as barplots where the bars represent the probabilities of each of the algorithms being better than the other, or the probability that they are practically equivalent. These plots are shown for the two metrics considered in the experiments: MAE and C-Index.

A.1 Comparison of the best model obtained with the rest of the algorithms

This section shows the results of the Bayesian tests that compare Med-k-NN with each of the other algorithms. The results are displayed in Figures 11-14. The simplex diagrams and posterior distribution barplots are shown for both MAE and C-Index metrics.

A.2 Comparison of distances within the same algorithm

This section shows the results of the Bayesian tests that compare both Euclidean distance and the distance learned by LM^3L for each of the classifiers used in the experiments. The results are shown in Figures 15-16. The simplex diagrams and posterior distribution barplots are shown for both MAE and C-Index metrics.

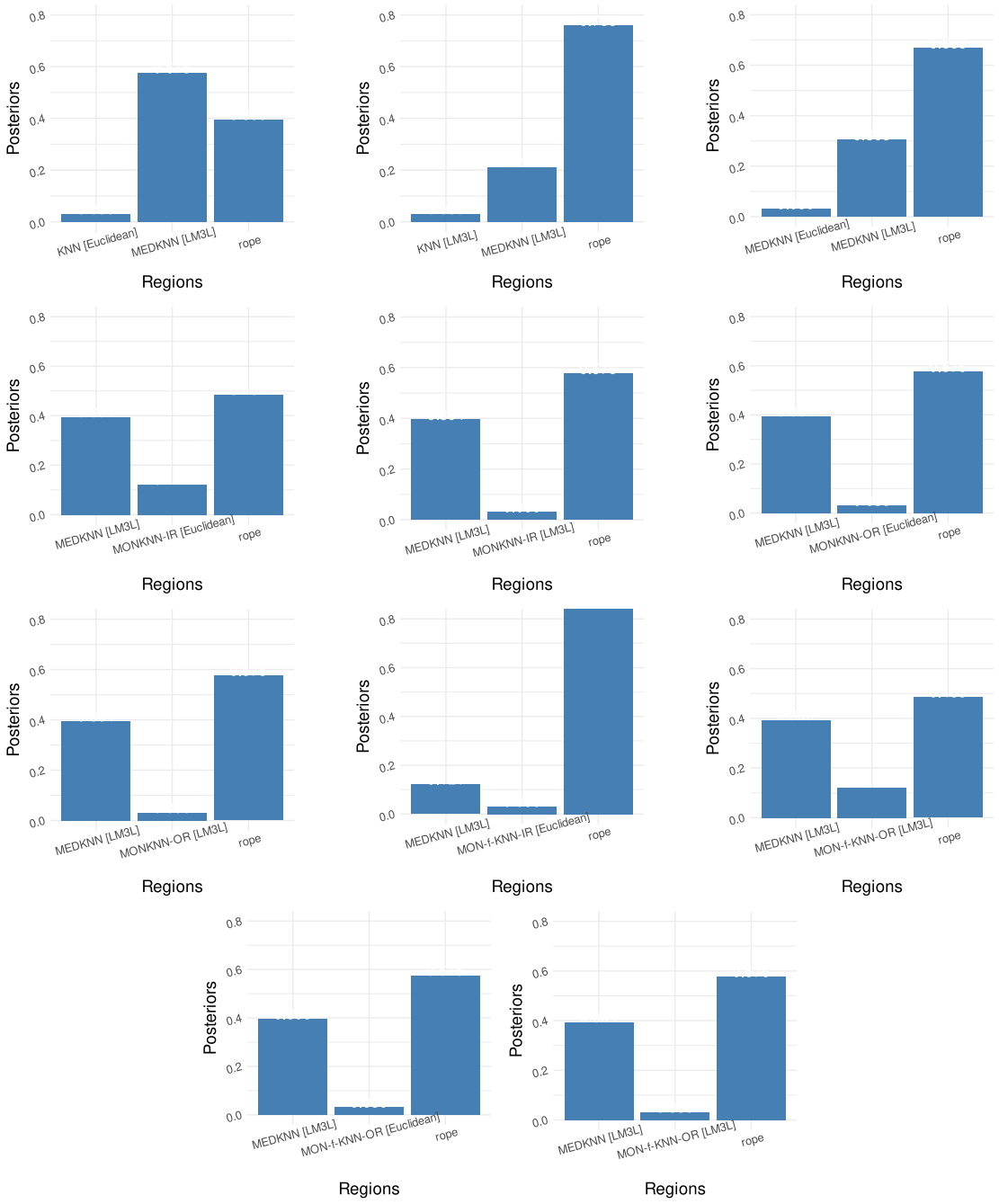


Figure 11: Posterior distributions using C-Index for the Bayesian comparison between the best model and the rest of the classifiers and distances.

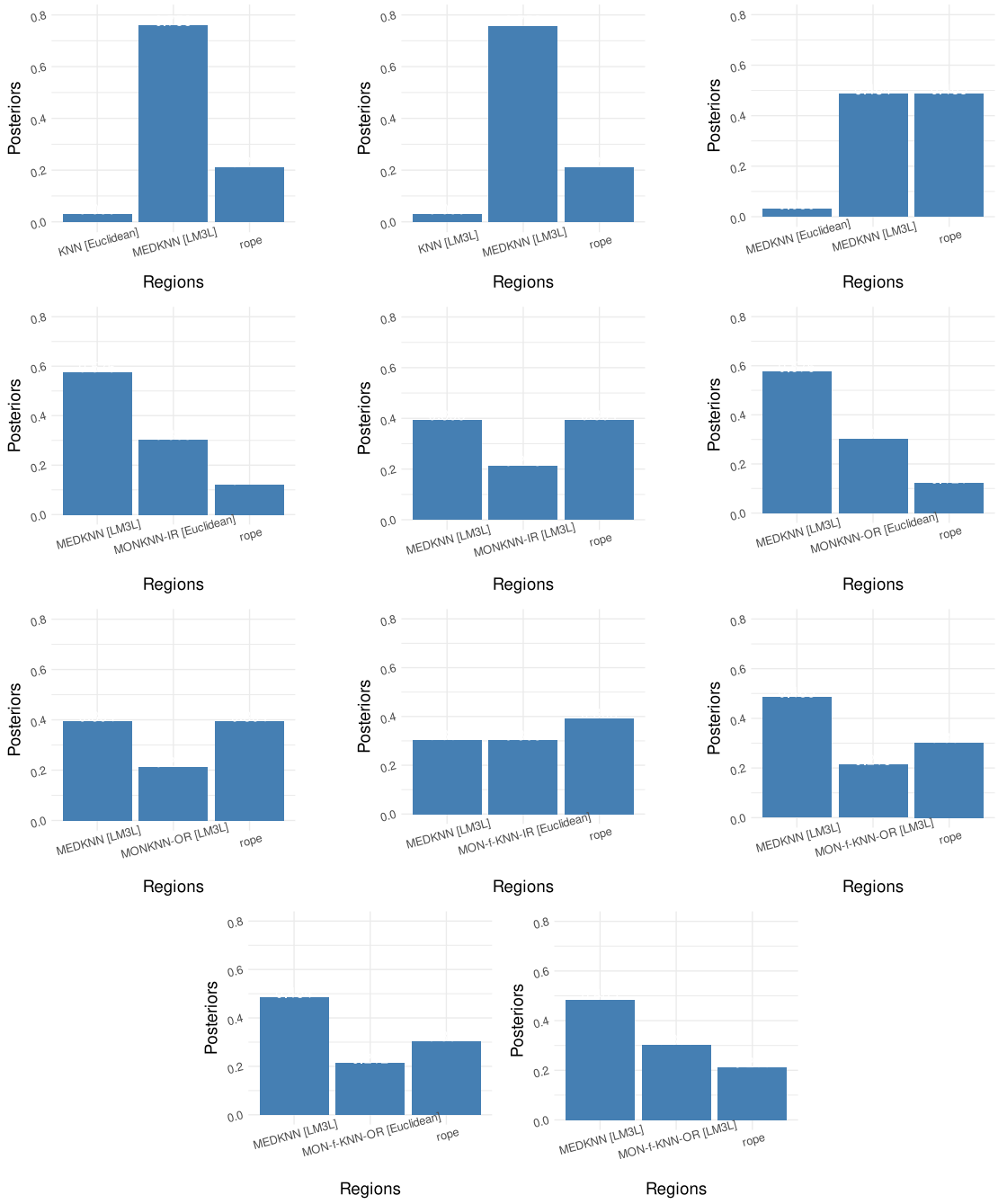


Figure 12: Posterior distributions using MAE for the Bayesian comparison between the best model and the rest of the classifiers and distances.

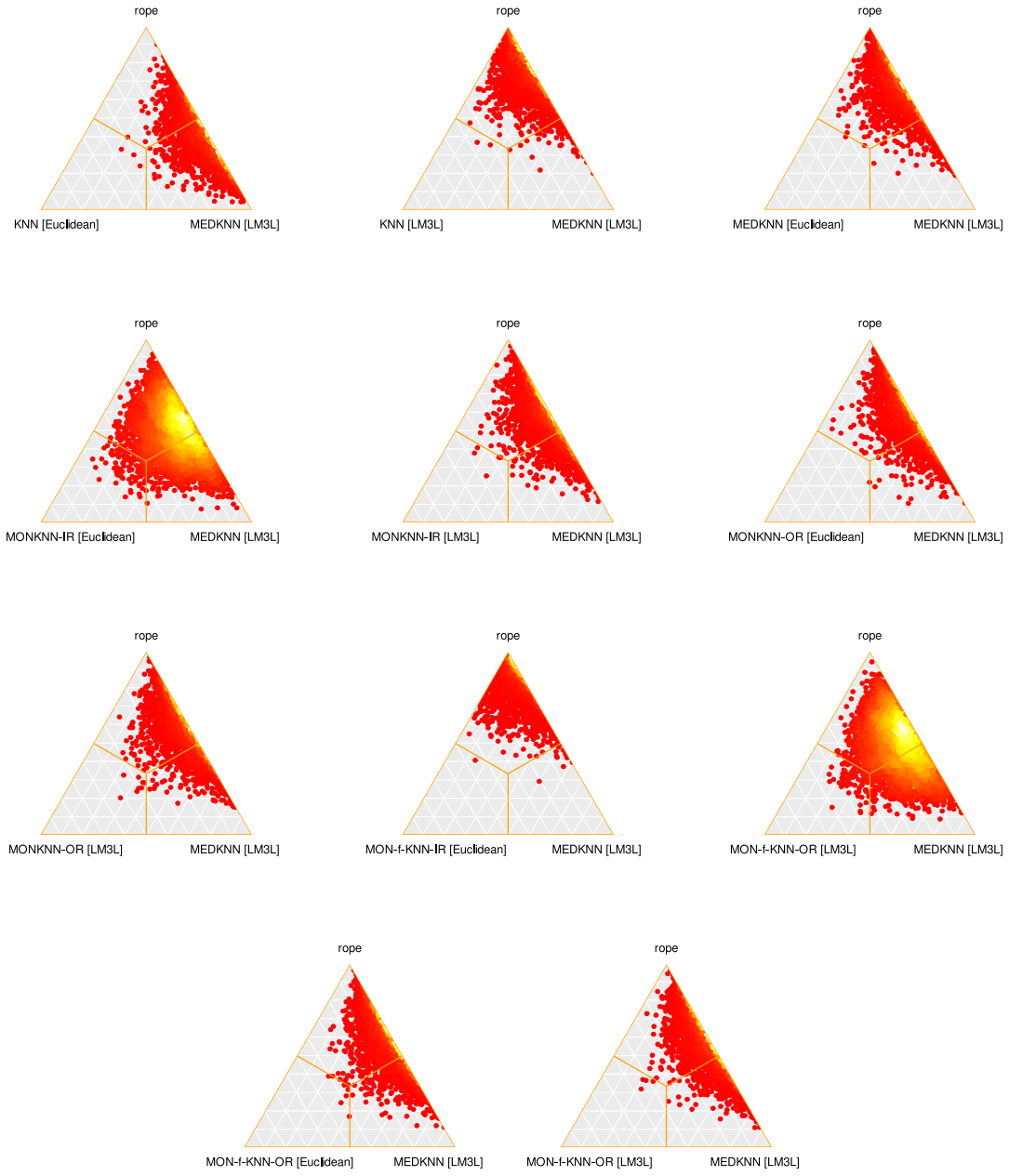


Figure 13: Simplex plots using C-Index for the Bayesian comparison between the best model and the rest of the classifiers and distances.

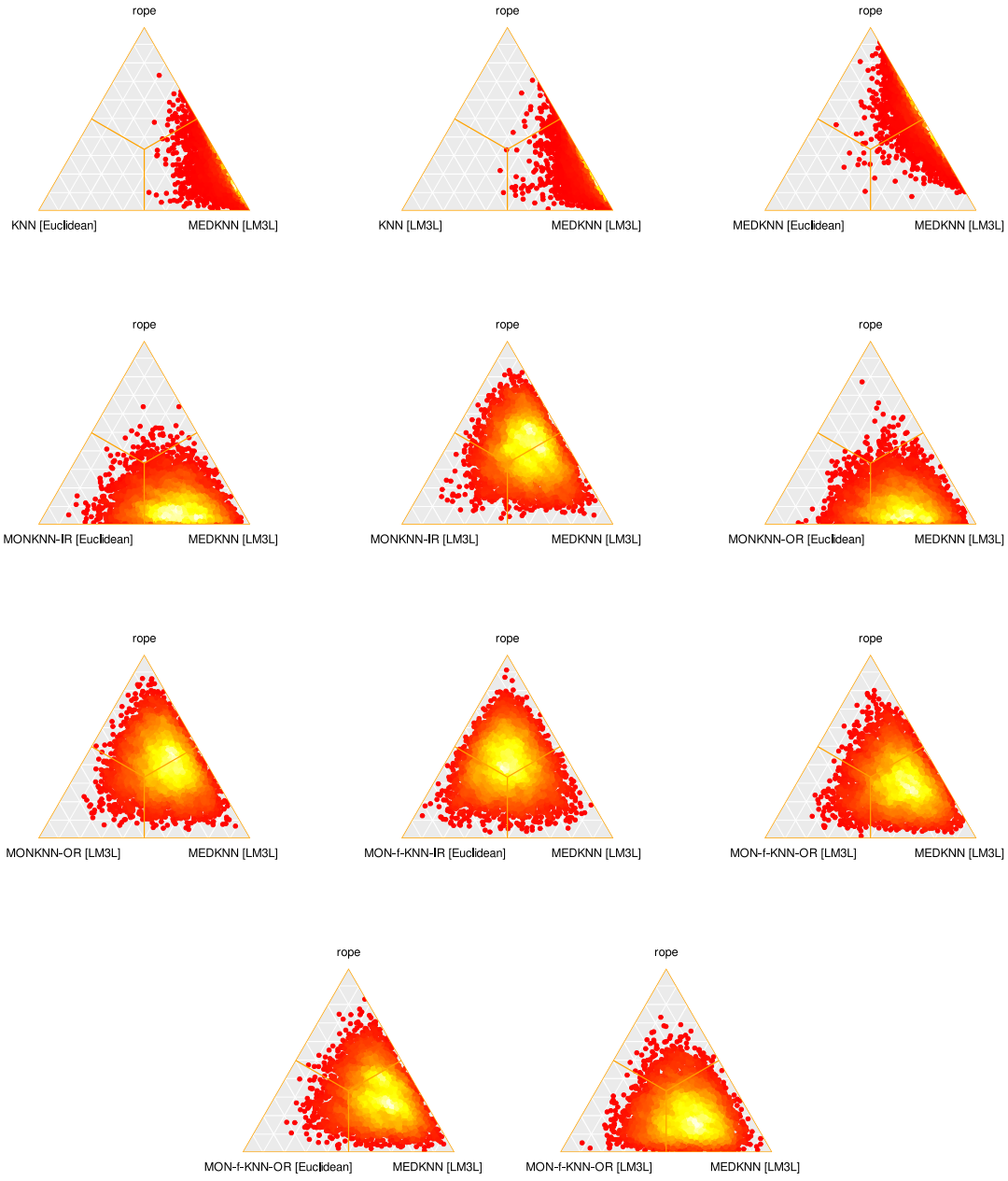


Figure 14: Simplex plots using MAE for the Bayesian comparison between the best model and the rest of the classifiers and distances.

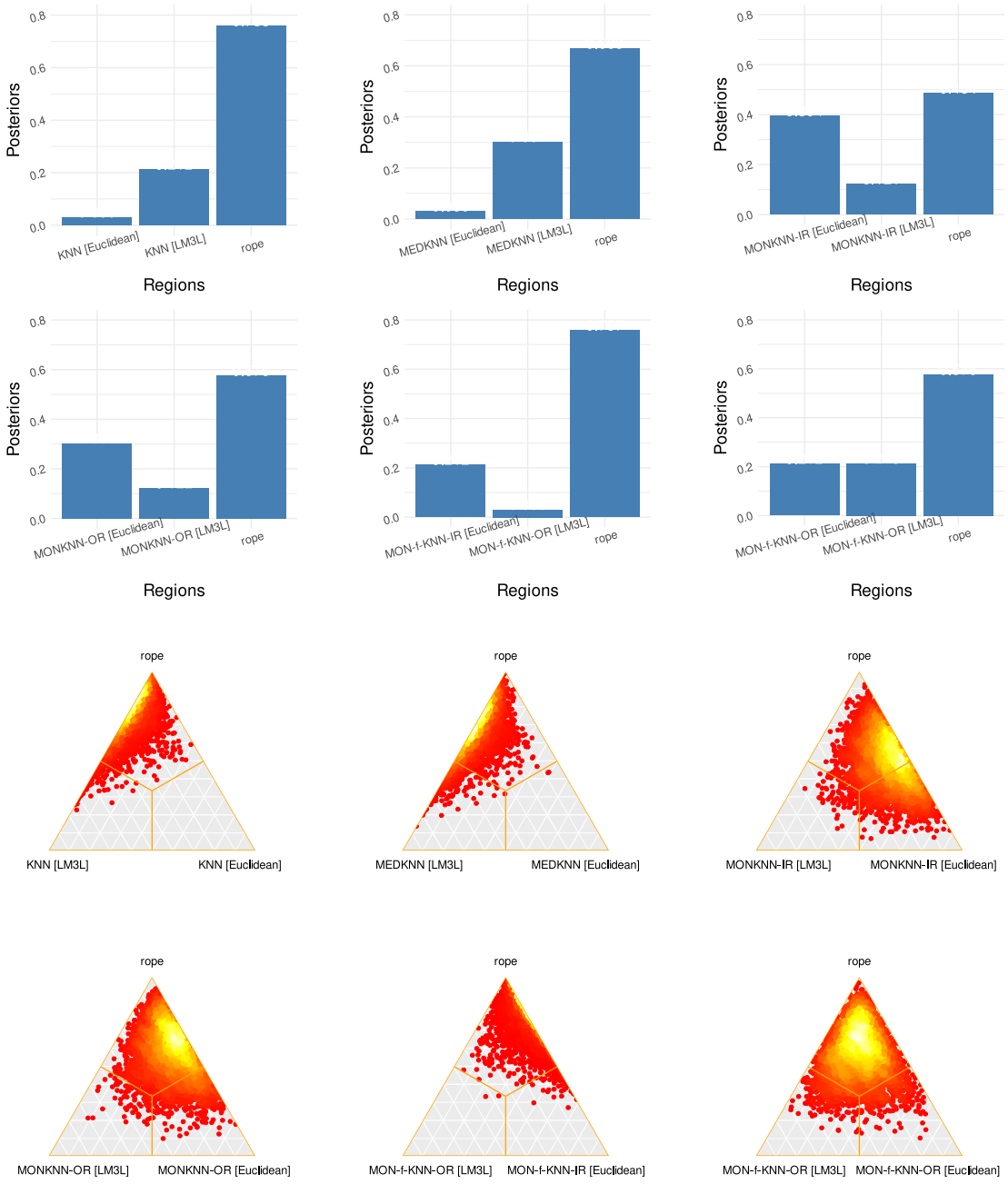


Figure 15: Simplex plots and posterior distributions using C-Index for the Bayesian comparison between LM^3L and the Euclidean distance for each classifier.

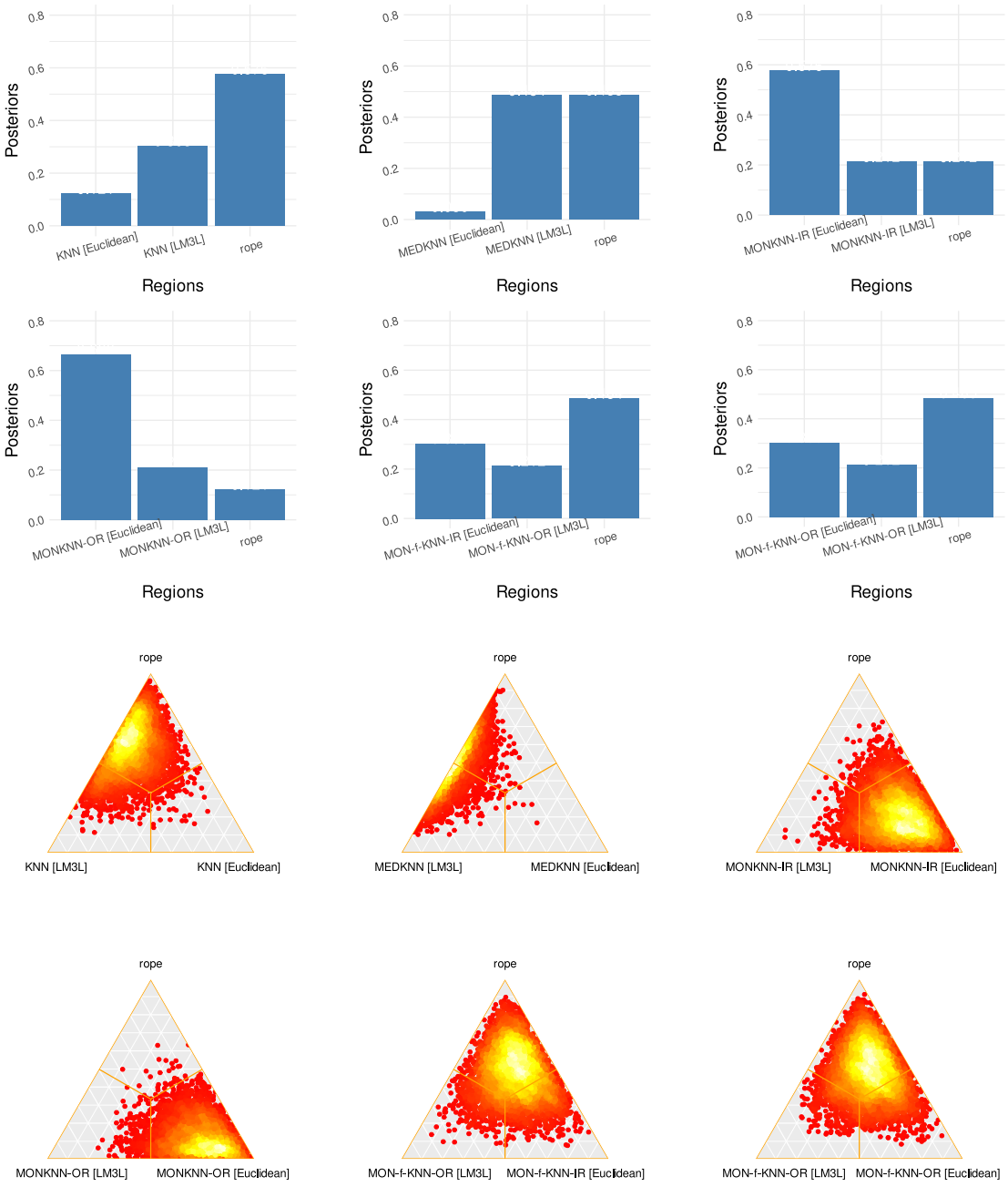


Figure 16: Simplex plots and posterior distributions using MAE for the Bayesian comparison between LM^3L and the Euclidean distance for each classifier.

References

[1] Wojciech Kotłowski and Roman Słowiński. Statistical approach to ordinal classification with monotonicity constraints. In *Preference learning ECML/PKDD 2008 workshop*, 2008.

[2] José-Ramón Cano, Pedro Antonio Gutiérrez, Bartosz Krawczyk, Michał Woźniak, and Salvador García. Monotonic classification: An overview on algorithms, performance measures and data sets. *Neurocomputing*, 341:168–182, 2019.

[3] Chih-Chuan Chen and Sheng-Tun Li. Credit rating with a monotonicity-constrained support vector machine model. *Expert Systems with Applications*, 41(16):7235–7247, 2014.

[4] Jose-Ramon Cano, Naif R Aljohani, Rabeeh Ayaz Abbasi, Jalal S Alowidbi, and Salvador Garcia. Prototype selection to improve monotonic nearest neighbor. *Engineering Applications of Artificial Intelligence*, 60:128–135, 2017.

[5] Jakob Schoeffer, Niklas Kuehl, and Yvette Machowski. "there is not enough information": On the effects of explanations on perceptions of informational fairness and trustworthiness in automated decision-making. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 1616–1628, 2022.

[6] David Leslie. Understanding artificial intelligence ethics and safety. *arXiv preprint arXiv:1906.05684*, 2019.

[7] Dangxing Chen and Weicheng Ye. Monotonic neural additive models: Pursuing regulated machine learning models for credit scoring. In *Proceedings of the Third ACM International Conference on AI in Finance*, pages 70–78, 2022.

[8] Pedro Antonio Gutiérrez, Maria Perez-Ortiz, Javier Sanchez-Monedero, Francisco Fernandez-Navarro, and Cesar Hervas-Martinez. Ordinal regression methods: survey and experimental study. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):127–146, 2015.

[9] Víctor Manuel Vargas, Pedro Antonio Gutiérrez, and César Hervás-Martínez. Unimodal regularisation based on beta distribution for deep ordinal regression. *Pattern Recognition*, 122:108310, 2022.

[10] Haiyan Chen, Yizhen Jia, Jiaming Ge, and Bin Gu. Incremental learning algorithm for large-scale semi-supervised ordinal regression. *Neural Networks*, 149:124–136, 2022.

[11] Víctor Manuel Vargas, Pedro Antonio Gutiérrez, Javier Barbero-Gómez, and César Hervás-Martínez. Soft labelling based on triangular distributions for ordinal classification. *Information Fusion*, 2023.

[12] Yuhua Qian, Hang Xu, Jiye Liang, Bing Liu, and Jieting Wang. Fusing monotonic decision trees. *IEEE Transactions on Knowledge and Data Engineering*, 27(10):2717–2728, 2015.

[13] Jieting Wang, Yuhua Qian, Feijiang Li, Jiye Liang, and Weiping Ding. Fusing fuzzy monotonic decision trees. *IEEE Transactions on Fuzzy Systems*, 28(5):887–900, 2019.

[14] Joao Marques-Silva, Thomas Gerspacher, Martin C Cooper, Alexey Ignatiev, and Nina Narodytska. Explanations for monotonic classifiers. In *International Conference on Machine Learning*, pages 7469–7479. PMLR, 2021.

[15] Thomas M Cover, Peter E Hart, et al. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[16] Wouter Duivesteijn and Ad Feelders. Nearest neighbour classification with monotonicity constraints. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 301–316. Springer, 2008.

[17] Sergio González, Salvador García, Sheng-Tun Li, Robert John, and Francisco Herrera. Fuzzy k-nearest neighbors with monotonicity constraints: Moving towards the robustness of monotonic noise. *Neurocomputing*, 439:106–121, 2021.

[18] Juan Luis Suárez, Salvador García, and Francisco Herrera. A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges. *Neurocomputing*, 425:300–322, 2021.

[19] Weiwei Liu, Donna Xu, Ivor W Tsang, and Wenjie Zhang. Metric learning for multi-output tasks. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):408–422, 2018.

[20] Zhongchen Ma and Songcan Chen. Multi-dimensional classification via a metric approach. *Neurocomputing*, 275:1121–1131, 2018.

[21] Juan Luis Suárez, Salvador García, and Francisco Herrera. Ordinal regression with explainable distance metric learning based on ordered sequences. *Machine Learning*, 110(10):2729–2762, 2021.

[22] Bac Nguyen, Carlos Morell, and Bernard De Baets. Distance metric learning for ordinal classification based on triplet constraints. *Knowledge-Based Systems*, 142:17–28, 2018.

[23] Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Ruslan R Salakhutdinov. Neighbourhood components analysis. In *Advances in neural information processing systems*, pages 513–520, 2005.

[24] Binbin Sang, Hongmei Chen, Lei Yang, Jihong Wan, Tianrui Li, and Weihua Xu. Feature selection considering multiple correlations based on soft fuzzy dominance rough sets for monotonic classification. *IEEE Transactions on Fuzzy Systems*, 30(12):5181–5195, 2022.

[25] Xiaoyan Zhang, Zongying Jiang, and Weihua Xu. Feature selection using a weighted method in interval-valued decision information systems. *Applied Intelligence*, pages 1–20, 2022.

[26] Abraham Berman and Robert J Plemmons. *Nonnegative matrices in the mathematical sciences*. SIAM, 1994.

[27] Juan Luis Suárez, Germán González-Almagro, Salvador García, and Francisco Herrera. A preliminary approach for using metric learning in monotonic classification. In *Advances and Trends in Artificial Intelligence. Theory and Practices in Artificial Intelligence: 35th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2022, Kitakyushu, Japan, July 19–22, 2022, Proceedings*, pages 773–784. Springer, 2022.

[28] John P Cunningham and Zoubin Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *The Journal of Machine Learning Research*, 16(1):2859–2900, 2015.

[29] Mengzi Tang, Raúl Pérez-Fernández, and Bernard De Baets. Fusing absolute and relative information for augmenting the method of nearest neighbors for ordinal classification. *Information Fusion*, 56:128–140, 2020.

[30] Jakob Schoeffer, Niklas Kuehl, and Isabel Valera. A ranking approach to fair classification. In *ACM SIGCAS Conference on Computing and Sustainable Societies*, pages 115–125, 2021.

[31] James M Keller, Michael R Gray, and James A Givens. A fuzzy k-nearest neighbor algorithm. *IEEE transactions on systems, man, and cybernetics*, (4):580–585, 1985.

[32] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[33] Fei Wang and Changshui Zhang. Feature extraction by maximizing the average neighborhood margin. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[34] Bac Nguyen, Carlos Morell, and Bernard De Baets. Supervised distance metric learning through maximization of the jeffrey divergence. *Pattern Recognition*, 64:215–225, 2017.

[35] Lorenzo Torresani and Kuang-chih Lee. Large margin component analysis. *Advances in neural information processing systems*, 19:1385, 2007.

[36] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.

[37] Tin Truong, Hai Duong, Bac Le, Philippe Fournier-Viger, and Unil Yun. Frequent high minimum average utility sequence mining with constraints in dynamic databases using efficient pruning strategies. *Applied Intelligence*, 52(6):6106–6128, 2022.

[38] Isaac Triguero, Sergio González, Jose M Moyano, Salvador García López, Jesús Alcalá Fernández, Julián Luengo Martín, Alberto Fernández Hilario, Jesús Díaz, Luciano Sánchez, Francisco Herrera, et al. Keel 3.0: an open source software for multi-stage analysis in data mining. In *International Journal of Computational Intelligence Systems*, pages 1238–1249. Atlantis Press, 2017.

[39] Arie Ben-David. Automatic generation of symbolic multiattribute ordinal knowledge-based dsss: methodology and applications. *Decision Sciences*, 23(6):1357–1372, 1992.

[40] Mithat Gönen and Glenn Heller. Concordance probability and discriminatory power in proportional hazards regression. *Biometrika*, 92(4):965–970, 2005.

[41] Juan Luis Suárez, Salvador García, and Francisco Herrera. pydml: A python library for distance metric learning. *Journal of Machine Learning Research*, 21(96):1–7, 2020.

[42] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research*, 18(1):2653–2688, 2017.

[43] Alessio Benavoli, Giorgio Corani, Francesca Mangili, Marco Zaffalon, and Fabrizio Ruggeri. A bayesian wilcoxon signed-rank test based on the dirichlet process. In *International conference on machine learning*, pages 1026–1034, 2014.

[44] Jacinto Carrasco, Salvador García, MM Rueda, Swagatam Das, and Francisco Herrera. Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review. *Swarm and Evolutionary Computation*, 54:100665, 2020.

Capítulo III

Trabajo en progreso: *Aprendizaje de distancias profundo con datos escasos aplicado al procesamiento del lenguaje natural*

«$e^{i\pi} + 1 = 0$».
– Leonhard Euler.

1 Introducción

En este capítulo se abordará el trabajo en progreso que se está realizando actualmente. Este trabajo aborda, dentro del procesamiento del lenguaje natural, la tarea de la extracción de relaciones. En particular, se considera esta tarea en un escenario de *few-shot learning*, en el que se dispone de escasos ejemplos para entrenar. Para afrontar este problema, se propone el uso de técnicas del aprendizaje de distancias profundo. El trabajo cubre así los objetivos de la tesis relacionados con el desarrollo de técnicas de aprendizaje profundo para problemas complejos y con la aplicación de los modelos desarrollados a problemas reales.

Este trabajo se estructura de la siguiente manera: a continuación, la Sección 1.1 introduce y motiva los problemas y herramientas con los que se trabajará en el resto de secciones. En la en la Sección 2 se proporciona la base necesaria para comprender el trabajo realizado. La Sección 3 describe con detalle la propuesta elaborada en este trabajo. En la Sección 4 se presentan los experimentos realizados y los resultados obtenidos. Finalmente, en la Sección 5 se exponen las conclusiones extraídas y las líneas futuras de investigación.

1.1 Motivación

La extracción de relaciones, una tarea fundamental en el procesamiento del lenguaje natural (PLN), desempeña un papel crucial en tareas como la extracción de información [SCYZ22], la población de bases de conocimiento [CYL+23] y los sistemas de preguntas y respuestas [CGZJ23]. Si bien se ha logrado un progreso significativo en la extracción de relaciones supervisada, el desafío de la extracción de relaciones *few-shot* (FSRE) sigue siendo una barrera. La FSRE implica identificar y clasificar relaciones entre entidades en un texto, incluso cuando el modelo tiene acceso a un número limitado de ejemplos de entrenamiento para cada relación. Los métodos tradicionales supervisados a menudo tienen dificultades en este escenario, ya que dependen en gran medida de la disponibilidad de datos etiquetados extensos.

En este contexto, el aprendizaje de distancias profundo [KB19] se ha destacado como una técnica prometedora para abordar el problema de FSRE. El aprendizaje de distancias profundo se centra en aprender representaciones que reflejen la estructura subyacente de similitud en los datos. La capacidad discriminativa de este enfoque destaca frente a los modelos tradicionales del aprendizaje profundo, erigiéndose el aprendizaje de distancias profundo como una herramienta poderosa cuando se disponen de pocos datos de entrenamiento.

Los modelos de lenguaje como BERT [DCLT18] y sus variantes han demostrado capacidades notables para comprender información contextual y mejorar tareas de PLN. Al integrar estrategias de aprendizaje de distancias profundo en estos modelos, se exploran enfoques que fomentan la similitud entre ejemplos con la misma relación y minimizan la similitud entre ejemplos de diferentes relaciones. En particular, el pre-entrenamiento contrastivo [PGH+20] en BERT junto con un aprendizaje final basado también en aprendizaje de distancias profundo, como es el caso de las redes de prototipos, ofrecen un camino promete-

dor para superar las limitaciones de la FSRE.

El modelo de aprendizaje para FSRE descrito anteriormente aún dispone de margen de mejora en muchos aspectos. En este trabajo, nos centramos en la perspectiva del aprendizaje de distancias, perfeccionando el modelo de dos formas diferentes:

- **Función de pérdida espejo.** En adición a la pérdida contrastiva base, se propone un nuevo término en la función de pérdida. Este término autosupervisado genera muestras sintéticas en las que se ignoran partes del texto. La finalidad es evitar que el modelo se focalice demasiado en partes del texto irrelevantes, para aumentar la capacidad de generalización.
- **Minería de muestreo robusta.** Se propone una nueva estrategia para muestrear los ejemplos con los que se pre-entrena el modelo. Puesto que esta supervisión distante puede contener ruido, se identifican y seleccionan ejemplos más fiables para el aprendizaje para tratar de disminuir el impacto del ruido. Esta identificación de ejemplos fiables se realiza iterativamente conforme avanza el pre-entrenamiento, y se basa en la calidad de los vecinos más cercanos de cada ejemplo.

El modelo desarrollado se evalua en el dataset *FewRel* [HZY+18, GHZ+19], un benchmark desarrollado específicamente para FSRE que contempla diferentes tareas de aprendizaje *few-shot*. En particular, nuestro modelo se prueba tanto en la tarea de *few-shot* de mismo dominio como en la *few-shot* con adaptación de dominio. Los resultados muestran que la propuesta mejora el rendimiento del modelo base en ambas tareas, lo que demuestra la eficacia de las mejoras propuestas.

2 Antecedentes y trabajo relacionado

En esta sección se describen los antecedentes necesarios para comprender el ámbito que abarca este trabajo, así como el trabajo relacionado que sirve de base para la propuesta realizada. Las Secciones 2.1, 2.2 y 2.3 describen los conceptos fundamentales de *few-shot learning*, extracción de relaciones y aprendizaje de distancias profundo, respectivamente. La Sección 2.4 describe el modelo de pre-entrenamiento contrastivo en BERT y el modelo de *few-shot learning* en el que se inspira la propuesta realizada.

2.1 Few-shot learning

Few-Shot Learning (FSL) [WYKN20] es un paradigma de aprendizaje automático que se centra en el aprendizaje de tareas con un número limitado de ejemplos de entrenamiento. Uno de los principales problemas en los modelos de aprendizaje tradicionales y de aprendizaje profundo en particular es que los modelos a menudo requieren grandes cantidades de datos para un aprendizaje efectivo, lo cual no es siempre factible en la práctica. En este contexto, el FSL se ha convertido en un área de investigación activa en los últimos años, con el objetivo de desarrollar modelos que puedan aprender de manera efectiva con pocos ejemplos de entrenamiento.

Formalmente, la descripción del problema de FSL asume la existencia de dos datasets: el dataset base, $\mathbb{D}_{\text{base}}$, un dataset auxiliar que no tiene que estar estrictamente relacionado con el problema con el que se desea trabajar, y el dataset novedoso, $\mathbb{D}_{\text{novel}}$, que suele ser pequeño y contiene los datos reales de nuestro problema. La estrategia suele ser utilizar $\mathbb{D}_{\text{base}}$ para aprender una representación general de los datos que sirva de punto de partida, y luego particularizar dicha representación con los datos de entrenamiento de $\mathbb{D}_{\text{novel}}$.

Del conjunto $\mathbb{D}_{\text{novel}}$ se extraen las tareas *few-shot* que se desean predecir. Se dice que una tarea es *C-way k-shot* si el conjunto de entrenamiento (denominado *conjunto soporte*) dispone de C clases y k ejemplos para cada clase. Dado un conjunto soporte, el objetivo es predecir con la mayor fiabilidad posible las clases de los ejemplos del conjunto de test (denominado *conjunto consulta*).

2.2 Extracción de relaciones

La extracción de relaciones (*relation extraction*, RE) [NJM21] es una tarea de PLN que consiste en extraer relaciones entre entidades en un texto. Por ejemplo, dada la oración “París es la capital de Francia” y las entidades “París” y “Francia”, el objetivo es identificar que “París” y “Francia” están relacionados por la relación “capital”.

Los sistemas de extracción de relaciones utilizan enfoques de procesamiento de lenguaje natural y aprendizaje automático para analizar grandes cantidades de texto y extraer información estructurada que puede ser utilizada en una amplia gama de aplicaciones. La precisión y el rendimiento de estos sistemas han mejorado significativamente en los últimos años gracias a avances en técnicas de aprendizaje profundo y el acceso a grandes conjuntos de datos anotados.

En particular, el enfoque de extracción de relaciones *few-shot* (FSRE) [QGXT20], ha ganado popularidad en los últimos años. Esta variante es especialmente útil cuando nos enfrentamos a relaciones poco frecuentes o nuevas, para las que no se dispone de muchos ejemplos de entrenamiento. Esto es habitual en dominios como el biomédico. En este contexto, se hace necesario unificar los conocimientos de las disciplinas de FSL y RE para desarrollar modelos que puedan aprender de manera efectiva a extraer relaciones con ejemplos de entrenamiento limitados.

2.3 Aprendizaje de distancias profundo

El aprendizaje de distancias profundo [KB19] engloba a un conjunto de modelos del aprendizaje profundo que se centran en aprender representaciones de los datos que permitan reflejar las similitudes entre datos de forma óptima, para poder realizar tareas de aprendizaje posteriores con mayor eficacia. Las técnicas de aprendizaje de distancias profundo presentan un mayor poder discriminativo que los modelos tradicionales de aprendizaje profundo, lo que las hace más apropiadas para tareas con pocos datos de entrenamiento, en las que los otros modelos suelen tener dificultades [LYMX23].

En los últimos años el aprendizaje de distancias profundo ha experimentado una evolución considerable, con la aparición de numerosos modelos. En particular, en este trabajo se consideran los dos siguientes modelos de interés:

- **Redes siamesas y pérdida contrastiva.** Las redes siamesas [KZS+15] fueron uno de los primeros modelos de aprendizaje de distancias profundo desarrollados. Originalmente, este modelo se concibe como una red neuronal con dos subredes idénticas que comparten los mismos parámetros. La red se alimenta con pares de ejemplos. Los pares positivos corresponden a ejemplos asociados a una misma clase, y los pares negativos corresponden a ejemplos de clases diferentes. La función de pérdida, denominada pérdida contrastiva, penaliza la distancia entre los pares positivos y bonifica la distancia entre los pares negativos [LKHS20]. De esta forma, la red aprende a generar representaciones que maximizan la similitud entre ejemplos de la misma clase y minimizan la similitud entre ejemplos de clases diferentes.

- **Redes de prototipos.** Las redes de prototipos [SSZ17] son modelos de aprendizaje de distancias profundo que aprenden prototipos para cada clase en el conjunto de entrenamiento. Estos prototipos se calculan a partir de las representaciones del conjunto de entrenamiento. Son un modelo especialmente útil en FSL, y utilizan una estrate-

gia de entrenamiento episódica [VBL+16] en la que, en cada episodio, se muestrea un problema diferente de *C-way k-shot* y un ejemplo de consulta en las condiciones del problema que se desea tratar. La función de pérdida se computa a partir de la probabilidad de que el ejemplo de consulta pertenezca a cada una de las C clases, para lo que se tiene en cuenta la similtud con respecto a cada prototipo.

2.4 Pre-entrenamiento contrastivo

El pre-entrenamiento contrastivo [RA23] se ha convertido en una función objetivo popular en muchas tareas del PLN, ya que permite capturar con mayor calidad las relaciones entre palabras, oraciones o entidades. La pérdida contrastiva fuerza a los modelos a aprender representaciones que colocan a los datos similares cerca en el espacio de atributos, mientras que los datos diferentes se colocan lejos. Esto conduce a representaciones semánticamente ricas, que capturan mejor la estructura subyacente y las relaciones entre los datos, y que proporcionan un punto de partida más sólido para las tareas de aprendizaje posteriores.

En los últimos años, el desarrollo de codificadores como BERT [DCLT18] ha supuesto un gran avance en el PLN. Estos modelos han demostrado una gran capacidad para entender el contexto, gracias a su entrenamiento bidireccional, lo que ha revolucionado la forma en que las máquinas comprenden el lenguaje humano. La versatilidad de estos modelos y su capacidad para transferir el conocimiento a multitud de tareas a través del pre-entrenamiento los convierten en una de las herramientas más poderosas de PLN en la actualidad.

En el problema de extracción de relaciones, el pre-entrenamiento contrastivo en BERT se ha utilizado con éxito para mejorar el rendimiento de los modelos de extracción de relaciones. Peng et al. [PGH+20] proponen por primera vez el pre-entrenamiento contrastivo para extracción de relaciones (CP) como se describe a continuación. Utilizando la misma estructura de transformer que BERT, se consideran dos funciones objetivo diferentes. Por un lado, se utiliza la misma pérdida de modelado de lenguaje enmascarado que BERT, que selecciona aleatoriamente tokens de entrada y los enmascara, y el modelo debe predecirlos. Denotaremos esta pérdida como $\mathcal{L}_{\text{MLM}}$.

Por otro lado, la pérdida contrastiva, $\mathcal{L}_{\text{CP}}$, seleccionará pares de oraciones de entrada y los acercará si hacen referencia a la misma relación, y los alejará si no. Dada una oración de entrada junto con las dos entidades a relacionar, que denominaremos respectivamente cabeza y cola, se utilizan los marcadores especiales `[E1]-[/E1]` y `[E2]-[/E2]` para rodear dichas entidades (p.e.: "`[CLS] [E1]` París `[/E1]` es la capital de `[E2]` Francia `[/E2] . [SEP]`"). Esta entrada se pasa por el codificador para obtener una representación de la oración. La representación de la cabeza y la cola se obtienen tomando la representación del token `[E1]` y `[E2]`, respectivamente. Finalmente, consideramos la representación concatenada de las entidades cabeza y cola, que se utilizará en la función de pérdida.

Dadas la representación positiva $(\mathbf{x}, \mathbf{x}^+)$ asociada a dos oraciones cuyas entidades tienen asociada la misma relación y las representaciones negativas $(\mathbf{x}, \mathbf{x}_\mathbf{i}^-)$, con $i = 1, \dots, N$, asocia-

das a oraciones cuyas entidades tienen asociadas relaciones diferentes, la pérdida contrastiva para el problema de RE se define como:

$$\mathcal{L}_{CP} = -\log \frac{\exp(\mathbf{x} \cdot \mathbf{x}^{+})}{\exp(\mathbf{x} \cdot \mathbf{x}^{+}) + \sum_{i=1}^{N} \exp(\mathbf{x} \cdot \mathbf{x}_{\mathbf{i}}^{-})}. \tag{1}$$

Para evitar la memorización de entidades por parte del modelo, aleatoriamente se reemplazan las entidades de la oración por el marcador especial `[BLANK]`, con una probabilidad $P_{\text{blank}} = 0{,}7$. El modelo y la función de pérdida anterior se aplican igualmente con las oraciones enmascaradas. Finalmente, la función de pérdida final del modelo de pre-entrenamiento se define como:

$$\mathcal{L} = \mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{CP}}. \tag{2}$$

El conjunto $\mathbb{D}_{\text{base}}$ de pre-entrenamiento utilizado consiste en un corpus de oraciones de Wikipedia en las que se mencionan pares de entidades que tienen una relación asociada en la base de conocimiento de Wikidata [VV19].

La aplicación al problema de extracción de relaciones *few-shot* se completa con un entrenamiento final, consistente en una red de prototipos, que se entrena por episodios con el dataset novedoso. La red de prototipos propuesta se diferencia de la red de prototipos genérica en que utiliza como representaciones la concatenación de las representaciones de las entidades descrita anteriormente, y el producto escalar en lugar de la distancia euclídea para medir la similitud entre los datos.

3 Propuesta

En esta sección se describe la propuesta elaborada actualmente para mejorar el modelo de pre-entrenamiento contrastivo en el problema de FSRE. Se desglosan las dos mejoras introducidas sobre el modelo base descrito anteriormente. La Sección 3.1 describe la nueva componente de la función de pérdida elaborada, y la Sección 3.2 describe la nueva estrategia de muestreo para la red durante el pre-entrenamiento.

3.1 Función de pérdida espejo

Aunque la pérdida contrastiva ha demostrado ser una estrategia exitosa para perfeccionar los pesos de BERT de cara a la extracción de relaciones, tiene algunos inconvenientes que merece la pena resaltar. Por un lado, al ser supervisada, el entrenamiento está limitado a las relaciones que estén presentes en la base de conocimiento, Wikidata, en este caso. Por otro lado, la supervisión distante puede contener ruido y en consecuencia hacer que la pérdida contrastiva acerque ejemplos que no deberían estar relacionados.

Para solventar estos problemas, siguiendo la estrategia auto-supervisada propuesta en otros problemas de PLN, como el de la representación palabras en contexto [LLC+21], se propone una nueva componente en la función de pérdida que genera muestras sintéticas en las que se ignoran partes del texto. De esta forma, se añaden nuevos ejemplos positivos no supervisados que no necesitan unirse mediante una relación existente en la base de conocimiento, y al ignorar partes del texto se evita que el modelo pueda sobreajustarse a partes del texto irrelevantes para extraer la relación.

El mecanismo de la función de pérdida propuesta, que denominaremos *pérdida espejo*, es análogo a la pérdida contrastiva supervisada, teniendo como principal diferencia el cómo se seleccionan los ejemplos positivos. Dado un batch de entrada para el modelo, para cada elemento del batch se decide aleatoriamente si enmascarar alguna parte del texto o no. Asumiendo que habitualmente la parte más informativa de la oración a la hora de identificar la relación se encuentra entre las dos entidades, se da mayor probabilidad a enmascarar las palabras que se encuentran antes de la primera entidad y después de la segunda. En caso de que se decida enmascarar alguna parte del texto, se sustituye por el marcador especial `[MASK]`. Los pares positivos se forman con las oraciones originales y su correspondiente versión enmascarada, mientras que los negativos engloban a una oración y el resto de oraciones no generadas a partir de ella en el batch. Dado una oración cuya representación asociada denotamos por $\mathbf{x}$, si llamamos $\mathbf{x}^{\mathbf{M}}$ a la representación de la oración enmascarada u oración espejo, la pérdida espejo se define análogamente a la pérdida contrastiva supervisada como:

$$\mathcal{L}_{\text{mirror}} = -\log \frac{\exp(\mathbf{x} \cdot \mathbf{x}^{\mathbf{M}})}{\sum_{\mathbf{x}' \neq \mathbf{x}} \exp(\mathbf{x} \cdot \mathbf{x}')}. \tag{3}$$

Y, finalmente, la función de pérdida final del modelo pre-entrenado se define como:

$$\mathcal{L} = \mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{CP}} + \mathcal{L}_{\text{mirror}}. \tag{4}$$

3.2 Muestreo robusto: identificando ejemplos fiables

Los métodos del aprendizaje profundo basados en tuplas, como es el caso de la pérdida contrastiva, dependen en gran medida de cómo se realice el muestreo de datos con los que se alimente a la red neuronal [XSP20]. Puesto que al considerar tuplas el número de posibilidades crece a números difíciles de manejar, es importante seleccionar las tuplas que puedan ser más relevantes para el aprendizaje, bien acelerando el proceso de aprendizaje o bien contribuyendo en mayor medida al aumento de la calidad del modelo.

En el caso del pre-entrenamiento contrastivo, el muestreo se realiza de forma aleatoria. Puesto que estos datos vienen de la supervisión distante del corpus de Wikipedia y la base de conocimiento de Wikidata, es bastante común que se incluyan ejemplos ruidosos o irrelevantes para el aprendizaje. En particular, hemos detectado tres situaciones que pueden darse con frecuencia:

- **Ejemplos cuyo vecindario está poblado con ejemplos referenciando a las mismas entidades.** Estos ejemplos pueden ser problemáticos por varios motivos. Por un lado, un vecindario así resulta poco informativo para el aprendizaje, puesto que está más enfocado a las entidades en sí que al contexto de la oración. Por otro lado, es más probable que entre tantas oraciones compartiendo entidades, haya vecinos que no estén haciendo referencia a la relación presente en el ejemplo, lo que puede tergiversar las similitudes que el modelo intenta aprender. Además, el muestreo de estos ejemplos quita opciones de ser muestreados a ejemplos con mayor calidad.
- **Ejemplos cuyo vecindario tiene pocos ejemplos referenciando a la relación.** Estos ejemplos no son convenientes ya que hay una probabilidad alta de que sean ruidosos. Por tanto, la función de pérdida contrastiva va a intentar acercarlo a ejemplos de la misma relación cuando sea muestreado, lo cual puede no ser beneficioso para el aprendizaje.
- **Ejemplos cuyo vecindario tiene la misma estructura de oración.** Estos ejemplos pueden constituir una cantidad abundante de los datos. Ejemplos de estos vecindarios pueden formarse debido a ejemplos que aparezcan repetidos en el conjunto de datos, o también formarse con oraciones del tipo: "X es el Y-ésimo pico más alto de Z", que forman grupos de oraciones poco habituales y que hacen referencia a contextos muy concretos. Estos ejemplos no son ruidosos en general, pero tampoco llegan a ser informativos por la poca variabilidad que presentan. Para identificar estos ejemplos, utilizamos el solapamiento de n-gramas normalizado promedio entre las oraciones del vecindario, para $n = 1, 2$.

De esta forma, se propone una nueva minería para muestrear los ejemplos con los que se pre-entrena el modelo. Dichos ejemplos, que llamaremos *ejemplos fiables*, serán aquellos que no verifiquen ninguna de las tres situaciones descritas anteriormente, fijados ciertos umbrales y tamaños de vecindario. Puesto que los vecindarios dependen de la distancia y de las representaciones de los datos, se propone ir seleccionando iterativamente los ejemplos fiables conforme avanza el pre-entrenamiento. Concretamente, cada 5 épocas de entrenamiento, se recalculan dichos ejemplos. También, para no perder toda la información presente en el resto de los datos, se muestrean también ejemplos del resto del conjunto, pero dando mayor peso al muestreo de los ejemplos fiables.

4 Experimentos

En esta sección se describen los experimentos realizados para evaluar la propuesta elaborada. En la Sección 4.1 se describe el setup experimental utilizado, y en la Sección 4.2 se presentan los resultados obtenidos.

4.1 Setup experimental

Para evaluar la propuesta elaborada, se utiliza el dataset *FewRel* [HZY+18, GHZ+19], un benchmark desarrollado específicamente para FSRE que contempla diferentes tareas de aprendizaje *few-shot*. El dataset está compuesto por varios subconjuntos. Por un lado, un subconjunto público, que es utilizado para entrenamiento y validación de los modelos. Por otro lado, un subconjunto privado, del que no se conocen las etiquetas, que es el que se utilizará para evaluar los modelos. Este subconjunto privado se presenta a modo de competición en la plataforma Codalab[1][2], donde se han de subir las predicciones para obtener los resultados finales.

El dataset público se construye alineando oraciones de Wikipedia con las relaciones en la base de conocimiento de Wikidata. Los oraciones y relaciones escogidas están seleccionadas de tal forma que no haya solapamiento entre *FewRel* y el entrenamiento distante seguido en el pre-entrenamiento contrastivo descrito anteriormente. El proceso de construcción del dataset pasa posteriormente por una anotación humana experta, en la que se filtran oraciones en las que no se puede deducir la relación entre las entidades elegidas.

El dataset público consta de ejemplos de 64 relaciones diferentes para entrenamiento y 16 para validación, con 100 instancias por cada relación. Los datasets privados disponen de relaciones diferentes cada uno, también con 100 instancias por relación. Del dataset privado se extraen 10000 problemas diferentes de C-way k-shot, cada uno con su conjunto soporte y con una instancia de consulta, con $C \in \{5, 10\}$ y $k \in \{1, 5\}$. El rendimiento de los modelos se mide según la tasa de acierto promedio sobre la relación de la instancia de consulta, para cada uno de los 10000 problemas privados del conjunto. Cada dataset privado cubre una tarea diferente. Dichos datasets y tareas se describen a continuación:

- **FewRel 1.0 [HZY+18].** Este dataset contiene 20 relaciones diferentes de la misma naturaleza que las presentes en el conjunto de entrenamiento y validación de *FewRel*.
- **FewRel 2.0 (adaptación de dominio) [GHZ+19].** Este dataset contiene 25 relaciones diferentes de naturaleza diferente a las presentes en el conjunto de entrenamiento y validación de *FewRel*. Concretamente, estos datos provienen del dominio biomédico,

[1] `https://codalab.lisn.upsaclay.fr/competitions/7395`
[2] `https://codalab.lisn.upsaclay.fr/competitions/7397`

y son obtenidas de la base de datos médica *PubMed*[3] y el grafo de conocimiento biomédico *UMLS*[4]. Este problema se convierte en mucho más desafiante que el de *FewRel 1.0*, puesto que durante el entrenamiento no se han visto datos similares.

- **FewRel 2.0 (ninguno-de-los-anteriores) [GHZ+19].** Este dataset extiende a *FewRel 1.0* incluyendo la posibilidad de que el ejemplo de consulta no tenga ninguna de las relaciones del conjunto soporte. En este caso, el modelo debe ser capaz de predecir la relación como la etiqueta adicional *"ninguno-de-los-anteriores"*. Se proporcionan diferentes problemas a evaluar, según la tasa de ejemplos de consulta que no pertenezcan a ninguna de las relaciones del conjunto soporte.

El modelo desarrollado, que denominaremos *CP-M-R* (*CP with Mirror loss and Reliable mining*) se ha comparado con el modelo base *CP* en los problemas *FewRel 1.0* y *FewRel 2.0* con adaptación de dominio. Ambos han sido pre-entrenados usando el corpus de Wikipedia y la base de conocimiento de Wikidata descritos en la Sección 2.4. Aunque la función de pérdida espejo de *CP-M-R* permite ampliar la cantidad de ejemplos de entrenamiento al no tener que depender de relaciones existentes en Wikidata, para realizar una comparación justa entre ambos modelos se ha optado por mantener el dataset en las mismas condiciones en ambos casos. Ambos han sido entrenados posteriormente con el dataset de entrenamiento de *FewRel* usando la red de prototipos. Ambos modelos se han entrenado durante 20 épocas.

En *CP-M-R*, para la función de pérdida espejo se ha utilizado una probabilidad de enmascarar $P_{\text{before}} = P_{\text{after}} = 0{,}3$ a las palabras de la oración tanto antes de la primera entidad como después de la segunda, y una probabilidad $P_{\text{middle}} = 0{,}15$ para enmascarar el texto entre ambas entidades. La actualización de ejemplos fiables se ha realizado 4 veces, cada 5 épocas de ejecución en el modelo. La proporción entre ejemplos fiables y ejemplos normales escogida ha sido de un par de ejemplos fiable por cada dos pares de ejemplos normales. Se ha escogido un tamaño de vecindario de 20 ejemplos, para identificar a los ejemplos fiables. Dentro del vecindario, se han elegido unos umbrales de 0.1, 0.5 y 0.9 para indicar si las entidades coincidentes en el vecindario son suficientemente pocas, si el número de relaciones comunes es lo suficiente alto o la proporción de oraciones con estructura similar es lo suficientemente baja, respectivamente. Para un vecino particular, se considera que tiene estructura demasiado similar al ejemplo de referencia si su solapamiento de n-gramas normalizado es mayor que 0.2. El resto de parámetros del modelo es común al del modelo CP [PGH+20].

4.2 Resultados

En las Tablas 1 y 2 se muestran los resultados obtenidos por los modelos *CP* y *CP-M-R* en los problemas *FewRel 1.0* y *FewRel 2.0* con adaptación de dominio, respectivamente. Se distinguen 4 escenarios de *few-shot* diferentes: *5-way 1-shot*, *5-way 5-shot*, *10-way 1-shot*

[3] `https://www.ncbi.nlm.nih.gov/pubmed/`

[4] `https://www.nlm.nih.gov/research/umls/`

y *10-way 5-shot*. En cada escenario, se muestra la tasa de acierto promedio sobre los 10000 problemas del dataset privado de *FewRel*. Se muestra también el resultado promedio de cada uno de los modelos.

Modelo	5-way 1-shot	5-way 5-shot	10-way 1-shot	10-way 5-shot	Promedio
CP	0.951	0.971	0.912	0.947	0.9452
CP-M-R	**0.961**	**0.975**	**0.926**	**0.964**	**0.956**

Cuadro 1: Resultados en FewRel 1.0

Modelo	5-way 1-shot	5-way 5-shot	10-way 1-shot	10-way 5-shot	Promedio
CP	0.797	0.849	**0.681**	0.798	0.781
CP-M-R	**0.799**	**0.894**	0.664	**0.805**	**0.790**

Cuadro 2: Resultados en FewRel 2.0 (adaptación de dominio)

En los resultados se puede observar que *CP-M-R* se impone en la mayoría de escenarios al modelo base *CP*, lo que demuestra la efectividad de las mejoras propuestas en este trabajo. En el problema *FewRel 1.0*, *CP-M-R* es capaz de imponerse en los 4 problemas. En *FewRel 2.0* con adaptación de dominio, se impone en 3 de los 4 problemas, destacando el escenario *5-way 5-shot*, en el que *CP-M-R* mejora en un 4.5 % al modelo base y roza el 90 % de acierto promedio.

5 Conclusiones

En este trabajo, se ha propuesto una estrategia de mejora para un modelo de *few-shot learning* de extracción de relaciones basado en pre-entrenamiento contrastivo. La estrategia propuesta, denominada *CP-M-R*, combina la pérdida espejo y la minería de ejemplos fiables para mejorar el rendimiento del modelo en tareas de extracción de relaciones *few-shot*, en particular en el escenario de adaptación de dominio.

Como futuras líneas de investigación, es interesante destacar el hecho de que el espacio de atributos resultante tras el pre-entrenamiento contrastivo puede no ser ideal para la tarea de estracción de relaciones, al poder quedar las relaciones separadas en múltiples clusters. Por ello, se propone como continuación de este trabajo la exploración de la distribución espacial de dichas representaciones, y el estudio particularizado de las representaciones de los ejemplos fiables. Y, si fuera necesario, la aplicación de técnicas de aprendizaje de distancias, tanto profundo como clásico [SGH21], tras el pre-entrenamiento sobre los ejemplos fiables para obtener un espacio de atributos más adecuado para el entrenamiento posterior.

Bibliography

[A+15] Aggarwal C. C. *et al.* (2015) *Data mining: the textbook*, volumen 1. Springer.

[AAES+23] Ali S., Abuhmed T., El-Sappagh S., Muhammad K., Alonso-Moral J. M., Confalonieri R., Guidotti R., Del Ser J., Díaz-Rodríguez N., and Herrera F. (2023) Explainable artificial intelligence (xai): What we know and what is left to attain trustworthy artificial intelligence. *Information Fusion* 99: 101805.

[ADRDS+20] Arrieta A. B., Díaz-Rodríguez N., Del Ser J., Bennetot A., Tabik S., Barbado A., García S., Gil-López S., Molina D., Benjamins R., *et al.* (2020) Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion* 58: 82–115.

[AFB23] Abnoosian K., Farnoosh R., and Behzadi M. H. (2023) Prediction of diabetes disease using an ensemble of machine learning multi-classifier models. *BMC bioinformatics* 24(1): 1–24.

[AM18] Akhtar N. and Mian A. (2018) Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access* 6: 14410–14430.

[AP88] Ashby F. G. and Perrin N. A. (1988) Toward a unified theory of similarity and recognition. *Psychological review* 95(1): 124.

[AP94] Aamodt A. and Plaza E. (1994) Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications* 7(1): 39–59.

[BB07] Bach N. and Badaskar S. (2007) A review of relation extraction. *Literature review for Language and Statistics II* 2: 1–15.

[BCDZ17] Benavoli A., Corani G., Demšar J., and Zaffalon M. (2017) Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research* 18(1): 2653–2688.

[BHS15] Bellet A., Habrard A., and Sebban M. (2015) *Metric learning*. Springer Nature.

[BKG23] Bank D., Koenigstein N., and Giryes R. (2023) Autoencoders. *Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook* pp. 353–374.

[Bur98] Burges C. J. (1998) A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery* 2(2): 121–167.

[CBHK02] Chawla N. V., Bowyer K. W., Hall L. O., and Kegelmeyer W. P. (2002) Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16: 321–357.

[CGK+19] Cano J.-R., Gutiérrez P. A., Krawczyk B., Woźniak M., and García S. (2019) Monotonic classification: An overview on algorithms, performance measures and data sets. *Neurocomputing* 341: 168–182.

[CGZJ23] Calle Gallego J. M. and Zapata Jaramillo C. M. (2023) Quare: towards a question-answering model for requirements elicitation. *Automated Software Engineering* 30(2): 25.

[CH67] Cover T. and Hart P. (1967) Nearest neighbor pattern classification. *IEEE transactions on information theory* 13(1): 21–27.

[Cho17] Chouldechova A. (2017) Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data* 5(2): 153–163.

[CK05] Chu W. and Keerthi S. S. (2005) New approaches to support vector ordinal regression. In *Proceedings of the 22nd international conference on Machine learning*, pp. 145–152.

[CK07] Chu W. and Keerthi S. S. (2007) Support vector ordinal regression. *Neural computation* 19(3): 792–815.

[CL14] Chen C.-C. and Li S.-T. (2014) Credit rating with a monotonicity-constrained support vector machine model. *Expert Systems with Applications* 41(16): 7235–7247.

[CMPT+17] Caelles S., Maninis K.-K., Pont-Tuset J., Leal-Taixé L., Cremers D., and Van Gool L. (2017) One-shot video object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 221–230.

[CYL+23] Chen Q., Yao H., Li S., Li X., Kang X., Lai W., and Kuang J. (2023) Fact-condition statements and super relation extraction for geothermic knowledge graphs construction. *Geoscience Frontiers* 14(5): 101412.

[DCLT18] Devlin J., Chang M.-W., Lee K., and Toutanova K. (2018) Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

[DCLT19] Devlin J., Chang M.-W., Lee K., and Toutanova K. (Junio 2019) BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota.

[DF08] Duivesteijn W. and Feelders A. (2008) Nearest neighbour classification with monotonicity constraints. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part I 19*, pp. 301–316. Springer.

[DH+06] Duda R. O., Hart P. E., *et al.* (2006) *Pattern classification*. John Wiley & Sons.

[DKJ+07] Davis J. V., Kulis B., Jain P., Sra S., and Dhillon I. S. (2007) Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine learning*, pp. 209–216. ACM.

[DRDSC+23] Díaz-Rodríguez N., Del Ser J., Coeckelbergh M., de Prado M. L., Herrera-Viedma E., and Herrera F. (2023) Connecting the dots in trustworthy artificial intelligence: From ai principles, ethics, and key requirements to responsible ai systems and regulation. *Information Fusion* page 101896.

[DS98] Draper N. R. and Smith H. (1998) *Applied regression analysis*, volumen 326. John Wiley & Sons.

[Dwo06] Dwork C. (2006) Differential privacy. In *International colloquium on automata, languages, and programming*, pp. 1–12. Springer.

[EFS23] Ehyaei A.-R., Farnadi G., and Samadi S. (2023) Causal fair metric: Bridging causality, individual fairness, and adversarial robustness. *arXiv preprint arXiv:2310.19391* .

[FDA23] Firdous N., Din N. M. U., and Assad A. (2023) An imbalanced classification approach for establishment of cause-effect relationship between heart-failure and pulmonary embolism using deep reinforcement learning. *Engineering Applications of Artificial Intelligence* 126: 107004.

[FGG+18] Fernández A., García S., Galar M., Prati R. C., Krawczyk B., and Herrera F. (2018) *Learning from imbalanced data sets*, volumen 10. Springer.

[GAPDP+23] González-Almagro G., Peralta D., De Poorter E., Cano J.-R., and García S. (2023) Semi-supervised constrained clustering: An in-depth overview, ranked taxonomy and future research directions. *arXiv preprint arXiv:2303.00522* .

[GBC17] Goodfellow I., Bengio Y., and Courville A. (2017) Deep learning. adaptive computation and machine learning. *Massachusetts, USA* .

[GGKC22] Ghojogh B., Ghodsi A., Karray F., and Crowley M. (2022) Spectral, probabilistic, and deep metric learning: Tutorial and survey. *arXiv preprint arXiv:2201.09267* .

[GGL+21] González S., García S., Li S.-T., John R., and Herrera F. (2021) Fuzzy k-nearest neighbors with monotonicity constraints: Moving towards the robustness of monotonic noise. *Neurocomputing* 439: 106–121.

[GHRS05] Goldberger J., Hinton G. E., Roweis S. T., and Salakhutdinov R. R. (2005) Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, pp. 513–520.

[GHZ+19] Gao T., Han X., Zhu H., Liu Z., Li P., Sun M., and Zhou J. (Noviembre 2019) FewRel 2.0: Towards more challenging few-shot relation classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 6250–6255. Association for Computational Linguistics, Hong Kong, China.

[GLH15] García S., Luengo J., and Herrera F. (2015) *Data preprocessing in data mining*. Springer.

[GPOSM+16] Gutierrez P. A., Perez-Ortiz M., Sanchez-Monedero J., Fernandez-Navarro F., and Hervas-Martinez C. (2016) Ordinal regression methods: survey and experimental study. *IEEE Transactions on Knowledge and Data Engineering* 28(1): 127–146.

[HA15] Hoffer E. and Ailon N. (2015) Deep metric learning using triplet network. In *Similarity-Based Pattern Recognition: Third International Workshop, SIMBAD 2015, Copenhagen, Denmark, October 12-14, 2015. Proceedings 3*, pp. 84–92. Springer.

[Har68] Hart P. (1968) The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory* 14(3): 515–516.

[HK06] Han J. and Kamber M. (2006) Data mining: Concepts and techniques, 2nd editionmorgan kaufmann publishers. *San Francisco, CA, USA* .

[HLS22] Hogan W., Li J., and Shang J. (Diciembre 2022) Fine-grained contrastive learning for relation extraction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 1083–1095. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates.

[HLW+16] Han L., Luo S., Wang H., Pan L., Ma X., and Zhang T. (2016) An intelligible risk stratification model based on pairwise and size constrained kmeans. *IEEE journal of biomedical and health informatics* 21(5): 1288–1296.

[HWL20] Halbersberg D., Wienreb M., and Lerner B. (2020) Joint maximization of accuracy and information for learning the structure of a bayesian network classifier. *Machine Learning* 109: 1039–1099.

[HWM+20] Huai M., Wang D., Miao C., Xu J., and Zhang A. (2020) Pairwise learning with differential privacy guarantees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volumen 34, pp. 694–701.

[HZY+18] Han X., Zhu H., Yu P., Wang Z., Yao Y., Liu Z., and Sun M. (Octubre-Noviembre 2018) FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4803–4809. Association for Computational Linguistics, Brussels, Belgium.

[Jol02] Jolliffe I. (2002) *Principal Component Analysis*. Springer Series in Statistics. Springer.

[KB19] Kaya M. and Bilge H. Ş. (2019) Deep metric learning: A survey. *Symmetry* 11(9): 1066.

[KLM96] Kaelbling L. P., Littman M. L., and Moore A. W. (1996) Reinforcement learning: A survey. *Journal of artificial intelligence research* 4: 237–285.

[KSVK23] Kennedy R. K., Salekshahrezaee Z., Villanustre F., and Khoshgoftaar T. M. (2023) Iterative cleaning and learning of big highly-imbalanced fraud data using unsupervised learning. *Journal of Big Data* 10(1): 106.

[KZS+15] Koch G., Zemel R., Salakhutdinov R., *et al.* (2015) Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volumen 2. Lille.

[LGGRG+20] Luengo J., García-Gil D., Ramírez-Gallego S., García S., and Herrera F. (2020) Big data preprocessing. *Cham: Springer* .

[LKHS20] Le-Khac P. H., Healy G., and Smeaton A. F. (2020) Contrastive representation learning: A framework and review. *Ieee Access* 8: 193907–193934.

[LL12] Lin H.-T. and Li L. (2012) Reduction from cost-sensitive ordinal ranking to weighted binary classification. *Neural Computation* 24(5): 1329–1367.

[LLC+21] Liu Q., Liu F., Collier N., Korhonen A., and Vulić I. (Noviembre 2021) MirrorWiC: On eliciting word-in-context representations from pretrained language models. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pp. 562–574. Association for Computational Linguistics, Online.

[LLSD20] Li Y., Li X., Sun Q., and Dong Q. (2020) Sar image classification using cnn embeddings and metric learning. *IEEE Geoscience and Remote Sensing Letters* 19: 1–5.

[LLW02] Lin Y., Lee Y., and Wahba G. (2002) Support vector machines for classification in nonstandard situations. *Machine learning* 46: 191–202.

[LPC+23] Liu L., Pei Z., Chen P., Luo H., Gao Z., Feng K., and Gan Z. (2023) An efficient gan-based multi-classification approach for financial time series volatility trend prediction. *International Journal of Computational Intelligence Systems* 16(1): 40.

[LSG+19] Lamy J.-B., Sekar B., Guezennec G., Bouaud J., and Séroussi B. (2019) Explainable artificial intelligence for breast cancer: A visual case-based reasoning approach. *Artificial intelligence in medicine* 94: 42–53.

[LYK+23] Lin Z., Yu S., Kuang Z., Pathak D., and Ramanan D. (2023) Multimodality helps unimodality: Cross-modal few-shot learning with multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19325–19337.

[LYMX23] Li X., Yang X., Ma Z., and Xue J.-H. (2023) Deep metric learning for few-shot image classification: A review of recent developments. *Pattern Recognition* page 109381.

[M+67] MacQueen J. *et al.* (1967) Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volumen 1, pp. 281–297. Oakland, CA, USA.

[Mir12] Mirkin B. (2012) *Clustering: a data recovery approach*. CRC Press.

[MR05] Maimon O. and Rokach L. (2005) Introduction to knowledge discovery in databases. In *Data mining and knowledge discovery handbook*, pp. 1–17. Springer.

[MVPC13] Mensink T., Verbeek J., Perronnin F., and Csurka G. (2013) Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(11): 2624–2637.

[MZY+19] Mao C., Zhong Z., Yang J., Vondrick C., and Ray B. (2019) Metric learning for adversarial robustness. *Advances in neural information processing systems* 32.

[NJM21] Nasar Z., Jaffry S. W., and Malik M. K. (2021) Named entity recognition and relation extraction: State-of-the-art. *ACM Computing Surveys (CSUR)* 54(1): 1–39.

[NMDB17] Nguyen B., Morell C., and De Baets B. (2017) Supervised distance metric learning through maximization of the jeffrey divergence. *Pattern Recognition* 64: 215–225.

[NMDB18] Nguyen B., Morell C., and De Baets B. (2018) Distance metric learning for ordinal classification based on triplet constraints. *Knowledge-Based Systems* 142: 17–28.

[PF91] Piateski G. and Frawley W. (1991) *Knowledge discovery in databases*. MIT press.

[PGH+20] Peng H., Gao T., Han X., Lin Y., Li P., Liu Z., Sun M., and Zhou J. (Noviembre 2020) Learning from Context or Names? An Empirical Study on Neural Relation Extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3661–3672. Association for Computational Linguistics, Online.

[PHY21] Park H., Hosseini H., and Yun S. (2021) Federated learning with metric loss. In *Workshop on Federated Learning for User Privacy and Data Confidentiality in ICML21*.

[Pri13] Price K. V. (2013) Differential evolution. In *Handbook of optimization: From classical to modern approach*, pp. 187–214. Springer.

[PVG+11] Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., *et al.* (2011) Scikit-learn: Machine learning in python. *the Journal of machine Learning research* 12: 2825–2830.

[QBL18] Qi H., Brown M., and Lowe D. G. (2018) Low-shot learning with imprinted weights. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5822–5830.

[QGXT20] Qu M., Gao T., Xhonneux L.-P., and Tang J. (2020) Few-shot relation extraction via bayesian meta-learning on relation graphs. In *International conference on machine learning*, pp. 7867–7876. PMLR.

[QLT+21] Qin Y., Lin Y., Takanobu R., Liu Z., Li P., Ji H., Huang M., Sun M., and Zhou J. (Agosto 2021) ERICA: Improving entity and relation understanding for pre-trained language models via contrastive learning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3350–3363. Association for Computational Linguistics, Online.

[RA23] Rethmeier N. and Augenstein I. (2023) A primer on contrastive pretraining in language processing: Methods, lessons learned, and perspectives. *ACM Computing Surveys* 55(10): 1–17.

[RL16] Ravi S. and Larochelle H. (2016) Optimization as a model for few-shot learning. In *International conference on learning representations*.

[SAM96] Sill J. and Abu-Mostafa Y. (1996) Monotonicity hints. *Advances in neural information processing systems* 9.

[SBB+16] Santoro A., Bartunov S., Botvinick M., Wierstra D., and Lillicrap T. (2016) Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pp. 1842–1850. PMLR.

[SC22] Sousa D. and Couto F. M. (2022) Biomedical relation extraction with knowledge graph-based recommendations. *IEEE Journal of Biomedical and Health Informatics* 26(8): 4207–4217.

[SCYZ22] Su X., Cheng C., Yang K., and Zhou X. (2022) A knowledge-based data augmentation framework for few-shot biomedical information extraction. In *China Health Information Processing Conference*, pp. 29–40. Springer.

[SE18] Satorras V. G. and Estrach J. B. (2018) Few-shot learning with graph neural networks. In *International conference on learning representations*.

[SGH21] Suárez J. L., García S., and Herrera F. (2021) A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges. *Neurocomputing* 425: 300–322.

[SLS+21] Summaira J., Li X., Shoib A. M., Li S., and Abdul J. (2021) Recent advances and trends in multimodal deep learning: a review. *arXiv preprint arXiv:2105.11087* .

[SLY+19] Shi Y., Li P., Yuan H., Miao J., and Niu L. (2019) Fast kernel extreme learning machine for ordinal regression. *Knowledge-Based Systems* 177: 44–54.

[SSBD14] Shalev-Shwartz S. and Ben-David S. (2014) *Understanding machine learning: From theory to algorithms*. Cambridge university press.

[SSZ17] Snell J., Swersky K., and Zemel R. (2017) Prototypical networks for few-shot learning. *Advances in neural information processing systems* 30.

[SYZ+18] Sung F., Yang Y., Zhang L., Xiang T., Torr P. H., and Hospedales T. M. (2018) Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1199–1208.

[TKS11] Taylor M. E., Kulis B., and Sha F. (2011) Metric learning for reinforcement learning agents. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 777–784.

[TL07] Torresani L. and Lee K.-C. (2007) Large margin component analysis. In *Advances in Neural Information Processing Systems*, pp. 1385–1392.

[TLL22] Tu J., Liu H., and Li C. (2022) Ordinal regression for direction-related anomaly detection. *IEEE Transactions on Neural Networks and Learning Systems* pp. 1–14.

[TMP+24] Triguero I., Molina D., Poyatos J., Del Ser J., and Herrera F. (2024) General purpose artificial intelligence systems (gpais): Properties, definition, taxonomy, societal implications and responsible governance. *Information Fusion* 103: 102135.

[VBL+16] Vinyals O., Blundell C., Lillicrap T., Wierstra D., *et al.* (2016) Matching networks for one shot learning. *Advances in neural information processing systems* 29.

[VEH20] Van Engelen J. E. and Hoos H. H. (2020) A survey on semi-supervised learning. *Machine Learning* 109(2): 373–440.

[VV19] Van Veen T. (2019) Wikidata. *Information technology and libraries* 38(2): 72–81.

[WKW12] Wang J., Kalousis A., and Woznica A. (2012) Parametric local metric learning for nearest neighbor classification. *Advances in Neural Information Processing Systems* 25.

[WS09] Weinberger K. Q. and Saul L. K. (2009) Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research* 10(Feb): 207–244.

[WT07] Weinberger K. Q. and Tesauro G. (2007) Metric learning for kernel regression. In *Artificial intelligence and statistics*, pp. 612–619. PMLR.

[WYKN20] Wang Y., Yao Q., Kwok J. T., and Ni L. M. (2020) Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)* 53(3): 1–34.

[WZ07] Wang F. and Zhang C. (2007) Feature extraction by maximizing the average neighborhood margin. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. IEEE.

[WZW+17] Wang J., Zhou F., Wen S., Liu X., and Lin Y. (2017) Deep metric learning with angular loss. In *Proceedings of the IEEE international conference on computer vision*, pp. 2593–2601.

[WZZ+23] Wang Y., Zhang Y., Zhu J., Liao W., Yuan M., and Zhou W. (2023) Enhancing conversational recommender systems via multi-level knowledge modeling with semantic relations. *Knowledge-Based Systems* 282: 111129.

[XJRN03] Xing E. P., Jordan M. I., Russell S. J., and Ng A. Y. (2003) Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pp. 521–528.

[XSP20] Xuan H., Stylianou A., and Pless R. (2020) Improved embeddings with easy positive triplet mining. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2474–2482.

[YPB+14] Yi X., Paulet R., Bertino E., Yi X., Paulet R., and Bertino E. (2014) *Homomorphic encryption*. Springer.

[YT19] Yu B. and Tao D. (2019) Deep metric learning with tuplet margin loss. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6490–6499.

[YTM+21] Yong C. W., Teo K., Murphy B. P., Hum Y. C., Tee Y. K., Xia K., and Lai K. W. (2021) Knee osteoarthritis severity classification with ordinal regression module. *Multimedia Tools and Applications* pp. 1–13.

[YWH+23] Yuan Y., Wei J., Huang H., Jiao W., Wang J., and Chen H. (2023) Review of resampling techniques for the treatment of imbalanced industrial data classification in equipment condition monitoring. *Engineering Applications of Artificial Intelligence* 126: 106911.

[ZBL+23] Zha D., Bhat Z. P., Lai K.-H., Yang F., and Hu X. (2023) Data-centric ai: Perspectives and challenges. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*, pp. 945–948. SIAM.

[ZFH23] Zhuang L., Fei H., and Hu P. (2023) Knowledge-enhanced event relation extraction via event ontology prompt. *Information Fusion* 100: 101919.

[ZG02] Zhu X. and Ghahramani Z. (2002) Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie Mellon University.

[ZG22] Zhao Y. and Gai Z. (2022) Transformation and optimization of rural ecological endowment industry chain based on constrained clustering algorithm. *Scientific Programming* 2022.

[ZHX+22] Zhang B., Hu Y., Xu D., Li M., and Li M. (2022) Skg-learning: a deep learning model for sentiment knowledge graph construction in social networks. *Neural Computing and Applications* 34(13): 11015–11034.

[ZLL+17] Zeng J., Liu Y., Leng B., Xiong Z., and Cheung Y.-m. (2017) Dimensionality reduction in multiple ordinal regression. *IEEE Transactions on Neural Networks and Learning Systems* 29(9): 4088–4101.

[ZLZ+21] Zhang Z., Lan C., Zeng W., Chen Z., and Chang S.-F. (2021) Beyond triplet loss: Meta prototypical n-tuple loss for person re-identification. *IEEE Transactions on Multimedia* 24: 4158–4169.

[ZXB+21] Zhang C., Xie Y., Bai H., Yu B., Li W., and Gao Y. (2021) A survey on federated learning. *Knowledge-Based Systems* 216: 106775.