Extensiones del Enfoque Genético Iterativo para el Aprendizaje de Reglas Difusas

Extensions of the Genetic Iterative Approach for Learning Fuzzy Rules

TESIS DOCTORAL

Programa Oficial de Doctorado en Tecnologías de la Información y la Comunicación

David García Muñoz

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Departamento de Ciencias de la Computación e Inteligencia Artificial

University of Granada, Spain

A thesis submitted in fulfilment of the requirements for the degree of Doctor of University of Granada

Granada, April 2015

Supervised by:
Antonio González Muñoz
Francisco G. Raúl Pérez Rodríguez

Editor: Universidad de Granada. Tesis Doctorales
Autor: David García Muñoz
ISBN: 978-84-9125-148-4
URI:http://hdl.handle.net/10481/40317

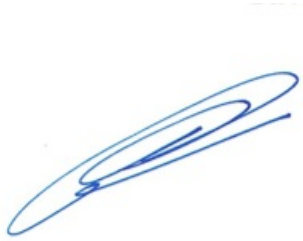
Tesis doctoral subvencionada por el programa predoctoral de Formación del Personal Docente e Investigador (FPDI) de la Junta de Andalucía (Resolución de 24 de febrero de 2010, BOJA n° 44, de 5 de marzo de 2010) y el proyecto adscrito a la misma P09-TIC-04813. También ha recibido subvención del proyecto TIN2012-38969 del Ministerio de Educación y Ciencia.

This work has been funded by the Andalusian Regional Government project P09-TIC-04813 and the Spanish MEC project TIN2012-38969.

El doctorando **David García Muñoz** y los directores de la tesis **Antonio González Muñoz** y **Francisco G. Raúl Pérez Rodríguez**. 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.

Granada a 17 de Abril de 2015.

Director/es de la Tesis



Fdo.: Antonio González Muñoz

Doctorando

Fdo.: David García Muñoz

Fdo.: Francisco G. Raúl Pérez Rodríguez

To my parents,
constant source of inspiration.

A mis padres,
continua fuente de inspiración.

Agradecimientos

Todo llega y todo pasa. Y durante ese tiempo, muchas son las personas que influyen en la consecución de nuestras metas, algunas de manera voluntaria y otras de forma involuntaria, pero ambas igualmente importantes.

En primer lugar, me gustaría agradecer especialmente a mis directores Antonio y Raúl, no sólo su exquisita profesionalidad, sino también su enorme paciencia, apoyo y calidez humana. Ni qué decir tiene, que sin ellos, esto no hubiese sido posible. Muchas gracias.

También quiero dar las gracias con todo mi cariño a mis padres. Han sido, son y seguirán siendo mis pies y mis manos. Jamás podré devolveros siquiera la mitad de todo lo que me habéis dado. Os quiero.

Agradezco igualmente a todos aquellos, que en distintos momentos me han ayudado a llevar a buen término este trabajo, con especial mención para John y Dimitris que tan buen trato y acogida me proporcionaron en Tesalónica. Y sin olvidarme, por supuesto, de mis compañeros Enrique, Yoel y Juan Carlos que estuvieron ahí siempre que los necesité.

Por último, pero no menos importante, agradecer a mi familia y amigos su comprensión y apoyo durante este tiempo de distancia.

Gracias a todos.

Contents

List of Figures

List of Tables

List of Tables

Chapter 1

Introduction

1.1 Motivation

According to the RAE (Real Academia Española de la Lengua), Artificial Intelligence (AI) is defined as "*the development and usage of computers with which the processes of human intelligence are reproduced*". John McCarthy, considered one of the pioneers in the Artificial Intelligence, first used this term in the Dartmouth Conference in 1956. In this conference, it was established the hypothesis that "*any learning aspect or any other feature of intelligence can be, in principle, described in such an accurately way that can be simulated with a machine*". One of the first definitions related to this topic came from Marvin Minsky, who belonged to the set of 10 scientists attendant to the conference. Minsky defined the AI as "*the science of building machines so that they are able to make tasks that, if were made by humans, it would be needed some intelligence*".

From that moment, the activity in this field has been very strong, arising several classic researching areas as learning. One of the main objectives of learning inside AI is to allow these systems to evolve depending on the experience acquired. Those intelligent systems having their learning capabilities limited or even containing all the knowledge inside, can repeat the same mistakes as many times as the situation occurs. In [80], Michalski deeply thinks about this idea, giving some definitions about learning, concluding in the following way: "*Learning is constructing or modifying representations of what is being experienced*".

One of the best known learning strategies is inductive learning. This is characterized by its ability to infer knowledge from the environment or an ex-

pert. Thus, in [78], Michalski distinguishes among two main classes of inductive learning: **concept learning from examples (concept acquisition)** and **concept learning from observation (descriptive generalization)**. In the first one, the objects (situations, processes) are pre-classified by an expert in one or more classes (concepts). The induced hypothesis can be seen as a concept recognition rule, that is, if an object satisfies the rule, then it represents the concept. In the descriptive generalization, the objective is to obtain a general description (law, theory) which characterize a set of observations. Both categories include a wide variety of problems [78].

Among the different methods of inductive learning and during the development of this work, the main efforts have been focused in one of the most well knowm methods, which is the inductive learning through rules. These rules usually represent "not evident" patterns (hidden patterns) existing in the original data. That is why they are an useful and powerful tool in data mining. Normally, they respond to the following structure:

$$\textit{if } (attribute_1, value_1) \textit{ and } (attribute_2, value_2) \textit{ and ... and}$$

$$(attribute_n, value_n) \textit{ then (class, value).}$$

The data from which these rules are obtained are often previously classified by an expert. That is the reason why this kind of learning is know as *supervised learning*.

Unfortunately, in real-world problems it is difficult to find structured data representing accurate situations. Even in our everyday language, terms like "very high" can represent different values depending on the problem domain, context, etc. So, the information is usually given in a imprecise or vague way. Fuzzy logic [119], provides good solutions when handling vagueness and uncertainty, due to its simplicity, flexibility and interpretability (similar to human thinking). Thus, in this framework it is represented by means of *fuzzy rules*, which are tools for managing this imprecise knowledge.

So, as it is explained in [26], "fuzzy *if/then* rules are rules whose antecedents, consequences or both are fuzzy rather than crisp". An example of this kind of rules is shown below:

IF $Petal\_Lenght$ is {VeryLow} and $Petal\_Width$ is {Medium} **THEN**

$$class \text{ is } IrisSetosa$$

where the fuzzy values for the variables $Petal\_Length$ and $Petal\_Width$ are $\{VeryLow, Low, Medium, High, VeryHigh\}$ and $class$ takes its values in a discrete domain in the set $\{IrisSetosa, IrisVersicolor, IrisVirginica\}$ (**Iris** database, UCI Machine Learning Repository [8]).

During the last years, many techniques have been developed to identify *fuzzy rule-based systems (FRBSs)*, some of which make use of ad-hoc methods, neural networks, genetic algorithms, etc.

Particularly, in this dissertation the idea of working with methods that try to obtain knowledge bases formed by such rules using *genetic algorithms (GAs)* is proposed. These methods are known in the literature with the name of *genetic fuzzy rule-based systems (GFRBSs)* [24]. The main characteristic of these systems is their capacity of learning using an evolutionary process in order to obtain the rule base describing the problem.

These rule bases or knowledge bases (KB) can be obtained using different strategies, being one of the most important those using an iterative approach, that is, the *iterative rule learning (IRL)* approach. The IRL approach is based on the *sequential covering (SC)* strategy [81], together with genetic algorithms for learning fuzzy rules. This model allows to extract in each iteration the best rule that will be part of the rule base later on.

A fuzzy rule learning algorithm using this idea is NSLV [46, 37]. This algorithm is an evolved version of the SLAVE algorithm [45, 37], which can be considered as the first fuzzy rule-based learning algorithm using the IRL approach. Thus, NSLV was developed with the objective of learning a set of interpretable and accurate fuzzy rules.

The SC strategy can be seen as an application of the hill climbing techniques to the learning problems. This strategy uses a greedy methodology in order to obtain the best rule after each iteration. This allows the use of an incremental model in which each step improves the previous solution by adding a new rule. However, this mechanism inherits one of the main drawbacks of the hill climbing methods, that is, the occurrence of a local optimum.

It is well-known that about hill climbing methods many proposals have been presented in order to avoid local optimums, some of which allow to improve the behavior of the original technique.

In classification problems employing the SC strategy it is commonly considered that a local optimum is reached if it is not possible to find a new useful rule when there exist examples not correctly classified by the already learned rules. This fact, in general, reduce the prediction capability and also interpretability to the knowledge extracted by the classification algorithm.

One of the techniques employed by the hill climbing methods for avoiding local optimums consist in extending the decision neighbourhood. The idea of this technique is to extend the locality of the decisions taking a farther view of the solutions.

An adaptation of this idea in classification problems by applying the SC strategy consist in taking into account the indirect relevance of the input attributes of the problem. This means trying to construct new attributes from the existing ones so that it is possible to describe in some other new way the examples of the training set.

On the other hand, the backtracking is a technique frequently used in searching problems in order to overcome local optimums and find new ways towards the solution. This technique recursively acts undoing the last operator applied and looks for a new path guided to the solution. The adaptation of this idea to the classification problems and, what is more, those based in the SC strategy is interesting because it introduces a new functionality in the learning process, that is, the decisions taken change from being irreversible to reversible. It also allows the review of the knowledge learned establishing new ways for finding better knowledge bases.

The purpose of this work is just taking the algorithm NSLV as a starting point in order to study some techniques similar to those previously exposed with the aim of extending the SC strategy. The development of different strategies in the SC tries to improve the rules extraction using basic criteria of accuracy and interpretability. In the next section a detailed description of these objectives is given.

1.2 Aims and Contributions

Thereby, the main objective of this thesis is to extend the iterative rule learning model for learning fuzzy rules so that, on the one hand, the indirect relevance of the input attributes is considered in the learning process and, on the other hand, the model used in NSLV is improved in order to review the knowledge extracted after each step.

In this way, the following sub-objectives (or partial objectives) have been proposed:

- Define a new model of rule and an inference model appropriate for handling the feature construction process.
- Define a filtering method for new attributes as well as the process of attribute selection used by the GA.
- Adapt the SC strategy so that the knowledge extracted after each iteration can be reviewed.
- Apply a pruning mechanism to limit the searching spaces of new solutions.

The main objective previously mentioned of extending the genetic iterative model is justified by the growing tendency of working with great amount of data, most of which are representative samples of bigger collections. These data are neither always uniform nor correctly structured. Even when they are classified by an expert they could be ambiguous. Because of this, last years, the learning algorithms and the fuzzy rule-based learning algorithms in particular, have been even more important. As the knowledge

extraction is more and more complex and necessary (as a consequence of the data complexity), the learning algorithms have evolved for going on being efficient.

The word "efficient" refers to a set of parameters different from each other. It can be referred when talking about the efficiency in time of an algorithm, the complexity of the knowledge extracted, the prediction capability, etc. Thus, the priority is to find a good trade off among all these measures which is one of the main problems to solve.

Particularly, the proposed techniques are focused in the study of the accuracy or prediction capability of the rule base, the time needed to obtain the model and the interpretability, this last one understood at rule base level (number of rules) and at a single rule level (number of conditions and closeness to natural language).

Improving the prediction capability of the rule base directly connects with the idea of considering the indirect relevance of the input attributes. As it was previously mentioned, sometimes the data are abundant but may be they are not very representative, that is, they give few information in order to establish a criterion for classification. One of the emerging techniques during the last years which allows working with datasets having these characteristics is the *feature construction*. This technique basically consists in creating new attributes from the combination of the original variables. The main contribution of the feature construction is the possibility of extracting additional and not trivial information apart from the single information associated to each variable.

This idea of constructing new attributes arises by means of the use of relations and functions in the antecedent of fuzzy rules. In addition to the improvement of the prediction capability, it is important to mention that an improvement of the interpretability of rules is also expected, as they should be closer to the use of natural language. However, the main disadvantages are expected to appear in relation to the increase of the complexity of the rule base and the time employed to obtain the model.

Opposite to this new situation in which the increase of the complexity of the rule base would reduce its interpretability, we connect with the second part of the objective, which is related to the extension of the SC strategy in order to review the knowledge extracted for being able to decide the way the rule base should be updated. This new model allows reducing the complexity of the rule base, the number of conditions per rule and the runtime. The main disadvantage of this strategy of knowledge review has to do with the dependency of the learning process to the appearance of general rules. These rules, which are difficult to replace, accelerate the learning of very specific rules with low influence when representing examples. For this reason, the appearance of very general rules should be delayed and the specificity of the final rules should also be limited. This is achieved by using pruned searching spaces through a completeness condition, which establishes

a covering threshold for each extracted rule.

It is important to mention that the knowledge review model provides the basic elements in order to evolve to a new model of incremental learning.

Next sections will describe in details the methods previously explained, focusing the main efforts in the changes made in the SC model of each algorithm.

1.3 Description of Chapters

This work is structured in seven chapters, including the current one, where it is introduced the general framework in which the proposed ideas underlie, together with the main motivations and goals.

Thereby, in Chapter 2, the problem of learning fuzzy rule-based systems is addressed. Moreover, the iterative rule learning approach (basic structure of our proposals), is described. Finally, some representative examples of learning methods based on genetic algorithms are presented.

Chapter 3 is devoted to describe **SLAVE**, a fuzzy rule-based learning algorithm characterized by the use of a genetic algorithm inside an iterative rule learning structure, as well as its evolutions: **SLAVE2** and **NSLV**. The main components of each one of the algorithms are deeply studied. This chapter ends with a detailed experimental section in which the most important results are summarized.

Chapter 4 aims to introduce the use of the *feature construction* technique under the genetic learning approach. In this sense, the feature construction has been developed through two different strategies: the use of relations in the antecedent of fuzzy rules and the use of functions. As a consequence, three different algorithms are presented: one for each strategy separately (**NSLV-R** and **NSLV-F**) and one more combining both strategies in the same learning method (**NSLV-FR**). After that, the results obtained by the different proposals together with a comparative study among them, are shown. Finally, the last section shows a real application of our feature construction method (NSLV-FR) to remotely sensed imagery. This last section results from the work developed during a three-months stay in the Aristotle University of Thessaloniki (Greece), under the supervision of Professor Dr. John B. Theocharis.

The application of the feature construction techniques usually results in more interpretable rule bases. Here, the term "interpretable" refers to an approximate model to natural language. However, the fact is that feature construction techniques normally increase the searching space, which translates in rule bases formed by higher number of rules and more conditions per rule and this is contrary to some interpretability measures. Due to this, in Chapter 5, a modified version of the iterative rule learning approach able to review the knowledge extracted, is proposed. The chapter also includes

an experimental study in which the proposed method (**NSLV-AR**) demonstrate a good performance when dealing with complex problems.

With the purpose of taking advantage of the strong points associated to the strategies used in the different proposals previously exposed, in Chapter 6 an integrated method combining feature construction and knowledge review, is defined (**SLAVE3**). Apart from that, three more sections take place in this chapter as part of the global experimental study. The first one is a full comparison between the algorithms proposed in the dissertation. The second one is devoted to compare SLAVE3 against other well known learning algorithms. The last section establishes the basic principles towards the definition of an incremental learning model.

Chapter 7 includes the conclusions derived from the results achieved by the different proposals described in this dissertation as well as their main contributions. It also contains some considerations in order to point out those potentially interesting elements for future developments.

1.4 Publications

This subsection will be devoted to enumerate the different publications in International Workshops, Conferences and Journals that have been carried out during the development of the research.

- García, D., González, A., Pérez, R., "A two-step approach of feature construction for a genetic learning algorithm", *IEEE International Conference on Fuzzy Systems*, 1255–1262 (2011, June).
- García, D., González, A., Pérez, R., "An iterative strategy for feature construction on a fuzzy rule-based learning algorithm", *11th International Conference on Intelligent Systems Design and Applications (ISDA)*, 1235–1240 (2011, November).
- García, D., González, A., Leyva, E., Pérez, R., "Un estudio experimental del uso de dominios con intensificaciones", *Congreso Español sobre Tecnologías y Lógica Fuzzy* (2012, February).
- García, D., González, A., Pérez, R., "A filter proposal for including feature construction in a genetic learning algorithm", *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, **20 (supp 02)**, 31–49 (2012, October).
- García, D., González A., Pérez R., "An empirical study about the behavior of a genetic learning algorithm on searching spaces pruned by a completeness condition", *Proceedings of the 2013 IEEE International Workshop on Genetic and Evolutionary Fuzzy Systems, GEFS 2013 - 2013 IEEE Symposium Series on Computational Intelligence, SSCI*, 8–15 (2013, April).

- García, D., González, A., Pérez, R., "A new iterative model to simplify the knowledge extracted on a fuzzy rule-based learning algorithm", *IEEE International Conference on Fuzzy Systems* (2013, July).
- García, D., González, A., Pérez, R., "Overview of the SLAVE learning algorithm: A review of its evolution and prospects", *International Journal of Computational Intelligence Systems*, **7 (6)**, 1194–1221 (2014, August).
- García, D., González, A., Pérez, R., "A feature construction approach for genetic iterative rule learning algorithm", *Journal of Computer and System Sciences*, **80(1)**, 101–117 (2014, December).
- García, D., Gámez, J. C., González, A., Pérez, R., "Using a sequential covering strategy for discovering fuzzy rules incrementally", *Proceedings of the IEEE International Conference on Fuzzy Systems* (2015, August).

1.5 Motivación

Según la Real Academia Española de la Lengua, la Inteligencia Artificial (IA) se define como "*el desarrollo y utilización de ordenadores con los que se intenta reproducir los procesos de inteligencia humana*". John McCarthy, considerado uno de los padres de la Inteligencia Artificial, acuñó este término en la Conferencia de Dartmouth en 1956. En ella, se establecía la conjetura de que "*cualquier aspecto del aprendizaje o cualquier otro rasgo de la inteligencia puede, en principio, ser descrito de manera tan precisa que puede construirse una máquina para simularlo*". Una de las primeras definiciones al respecto vino de la mano de Marvin Minsky, también perteneciente al grupo de 10 científicos asistentes a la conferencia. Según Minsky la IA "*es la ciencia de construir máquinas para que hagan cosas que, si las hicieran los humanos, requerirían inteligencia*".

A partir de entonces, la actividad en este campo ha sido muy intensa, dando lugar a diversas áreas de investigación clásicas entre las que se encuentra el aprendizaje. Uno de los principales objetivos del aprendizaje dentro de la inteligencia artificial es permitir a estos sistemas evolucionar en función de la experiencia adquirida. Aquellos sistemas inteligentes que tienen limitadas sus propiedades de aprendizaje o bien contienen todo el conocimiento programado en ellos, pueden repetir los mismos errores tantas veces como se genere la situación en cuestión. Michalski en [80], hace una profunda reflexión sobre esta idea y aporta diferentes definiciones de aprendizaje, concluyendo de la siguiente manera: "*Aprender es construir o modificar representaciones de aquello que está ocurriendo*".

Una de las estrategias de aprendizaje mas conocidas es la de aprendizaje inductivo. Ésta se caracteriza por su capacidad de inferir conocimiento a partir del entorno o del conocimiento experto. Así, en [78], Michalski distingue entre dos grandes tipos de aprendizaje inductivo: **aprendizaje a partir de ejemplos (adquisición de conceptos)** y **aprendizaje a partir de la observación (generalización descriptiva)**. En el primero de ellos, los objetos (situaciones, procesos, etc) son preclasificados por un experto, en una o más clases (conceptos). La hipótesis de inducción se puede ver como una regla de reconocimiento conceptual, esto es, si un objeto satisface la regla, entonces ésta representa el concepto. En la generalización descriptiva, el objetivo es obtener una descripción general (ley, teoría), que caracterice una colección de observaciones. Ambas categorías engloban una amplia variedad de problemas [78].

Entre los diferentes métodos de aprendizaje inductivo y durante el desarrollo de esta tesis, nos hemos centrado en uno de los más conocidos, esto es, el aprendizaje inductivo por medio de reglas. Estas reglas suelen expresar de forma natural e intuitiva patrones "no evidentes" existentes en los datos. Es por esto que conforman una herramienta muy útil en la minería de datos. Habitualmente, éstas se representan de la forma:

if $(attribute_1, value_1)$ *and* $(attribute_2, value_2)$ *and ... and*

$(attribute_n, value_n)$ *then (class, value).*

Los datos a partir de los cuales se obtienen estas reglas inductivas son, por lo general, previamente clasificados por un experto. De ahí que este tipo de aprendizaje se conozca con el nombre de *aprendizaje supervisado*.

Lamentablemente, en problemas del mundo real, no es fácil encontrar datos tan perfectamente estructurados o tales que representen situaciones exactas. Incluso en nuestro lenguaje natural, términos como "muy alto" pueden representar valores muy diferentes dependiendo del dominio del problema, contexto en el que se ubique, etc. Es decir, es muy habitual que la información de la que se dispone sea vaga o imprecisa. La lógica difusa [119] aporta soluciones a la hora de manejar esta incertidumbre e imprecisión, debido principalmente a su simplicidad, felxibilidad e interpretabilidad (proximidad al razonamiento humano). Así, en el marco en el que nos estamos situando, esto viene representado por el uso de *reglas difusas*, que van a servir como herramienta para representar conocimiento impreciso y/o incierto.

De esta forma, tal como se explica en [26], "las reglas difusas del tipo *si/entonces* son reglas donde bien el antecedente, el consecuente o ambos son difusos en lugar de crisp". Un ejemplo de este tipo de reglas se muestra a continuación:

IF $Petal\_Lenght$ is {VeryLow} and $Petal\_Width$ is {Medium} **THEN**

$class$ is $IrisSetosa$

donde los valores difusos para las variables $Petal\_Length$ y $Petal\_Width$ son $\{VeryLow, Low, Medium, High, VeryHigh\}$ y $class$ toma sus valores en un dominio discreto en el conjunto $\{IrisSetosa, IrisVersicolor, IrisVirginica\}$ (**Iris** database, UCI Machine Learning Repository [8]).

En los últimos aos, se han desarrollado diferentes técnicas que permiten identificar *sistemas basados en reglas difusas (FRBS)*, algunas de las cuales usan métodos ad-hoc, redes neuronales, algoritmos genéticos, etc.

En particular, en esta tesis se plantea la idea de trabajar con métodos que buscan obtener bases de conocimiento formadas por este tipo de reglas usando algoritmos genéticos (GAs). Por tanto, se hace uso de lo que en la literatura aparece con el nombre de *genetic fuzzy rule-based systems (GFRBS)* [24]. La principal característica de estos sistemas es su capacidad de aprender por medio de un proceso evolutivo, obteniendo la base de reglas que describe el problema.

Estas bases de reglas o de conocimiento (KB) se pueden generar con diferentes estrategias, siendo una de las más reconocidas aquella que usa un enfoque iterativo (*enfoque de aprendizaje de reglas iterativo*, en inglés IRL). Éste está basado en el uso del estrategia de recubrimiento secuencial (SC) [81], junto con algoritmos genéticos para el aprendizaje de reglas difusas. Dicho modelo permite extraer en cada iteración la mejor regla que posteriormente pasará a formar parte de la base de reglas.

Un algoritmo de aprendizaje de reglas basado en estas ideas es NSLV [46, 37]. Este algoritmo es la evolución del algoritmo SLAVE [45, 37], que puede considerarse como el primer algoritmo de aprendizaje de reglas difusas basado en el enfoque iterativo de aprendizaje de reglas. Así, NSLV fue desarrollado con la idea de aprender un conjunto de reglas difusas interpretables y con un alto grado de precisión.

La estrategia de recubrimiento secuencial puede verse como una aplicación de las técnicas de escalada a los problemas de aprendizaje. Esta estrategia hace uso de una metodología "greedy" para obtener la mejor regla en cada iteración. Esto permite el uso de un modelo incremental en el que en cada paso se mejora la solución previa con la adición de una nueva regla. Sin embargo, también se hereda una de las principales deficiencias de los métodos de escalada, es decir, se corre el riesgo de que el proceso se quede bloqueado por alcanzar un óptimo local.

Es conocido que sobre los métodos de escalada se han presentado diferentes propuestas para intentar salir de los óptimos locales, dando lugar a nuevas técnicas derivadas que permiten mejorar el comportamiento de la técnica original.

En los problemas de clasificación que usan la estrategia de recubrimiento secuencial, se puede entender que se ha alcanzado un óptimo local cuando no es posible encontrar una nueva regla útil existiendo aún ejemplos que no son correctamente clasificados por las reglas ya aprendidas. Este hecho, en general, resta capacidad de predicción e interpretabilidad al conocimiento extraído por el algoritmo de clasificación.

Una de las técnicas usadas por los métodos basados en ascensión de colinas para evitar los óptimos locales consiste en ampliar el vecindario de decisión. La idea de esta técnica es extender la localidad de los caminos de decisión, tomando una visión de las soluciones sobre un horizonte más lejano. Una adaptación de esta técnica de ampliación de vecindario al problema de la clasificación usando recubrimiento secuencial consiste en tener en cuenta la relevancia indirecta de los atributos de entrada al problema. Esto es, intentar construir nuevos atributos a partir de los ya existentes que permitan establecer una nueva forma de explicar los ejemplos del conjunto de entrenamiento.

Por otra parte, el "backtracking" es una técnica bastante usada en problemas de búsqueda para salir de óptimos locales y buscar nuevos caminos hacia la solución. Esta técnica consiste en deshacer de forma recursiva el

último operador aplicado y buscar un nuevo camino hacia la solución. La adaptación de esta idea a los problemas de clasificación, y en especial a aquellos basados en SC strategy, es interesante ya que introduce una funcionalidad valiosa en el proceso de aprendizaje, esto es, las decisiones tomadas pasan de tener caracter irrevocable a tener caracter revocable. La aplicación de una idea equivalente a la de backtraking en un algoritmo de clasificación otorga a las decisiones el caracter de revocable, y por consiguiente, permite revisar el conocimiento aprendido hasta el momento y establecer caminos alternativos para encontrar mejores bases de conocimiento.

La idea de esta tesis es precisamente tomar el algoritmo NSLV como punto de partida y estudiar la aplicación de técnicas, semejantes a las que se acaban de describir, para adaptar su estrategia de recubrimiento secuencial con la intención de mejorar la capacidad de extracción de reglas del mismo con criterios básicos de precisión e interpretabilidad. En la siguiente sección se hace una descripción más detallada de estos objetivos.

1.6 Objetivos y Aportaciones

De esta forma, el objetivo fundamental de la tesis es extender el modelo genético iterativo para el aprendizaje de reglas difusas de manera que, por un lado, se incorpore al proceso de aprendizaje la relevancia indirecta de los atributos de entrada, y por otro, sea capaz de mejorar el modelo empleado por NSLV de forma que dotemos al sistema de la capacidad de revisar el conocimiento extraído en cada paso.

Para ello, se han planteado los siguientes subobjetivos (u objetivos parciales):

- Definir un nuevo modelo de regla y un modelo de inferencia apropiados para gestionar el proceso de construcción de atributos.
- Definir un mecanismo de filtrado de nuevos atributos, así como el proceso de selección de atributos usado por el algoritmo genético.
- Adaptar el SC strategy de forma que permita revisar el conocimiento extraído en cada iteración.
- Aplicar un mecanismo de poda que acote los espacios de búsqueda de nuevas soluciones.

Este objetivo principal de extender el modelo genético iterativo, lo justifica en parte la creciente tendencia a trabajar con cantidades ingentes de datos, muchos de ellos muestras representativas de colecciones aún mayores. Estos datos no siempre son homogéneos ni están perfectamente estructurados e incluso a la hora de ser clasificados por un experto, podrían presentar ambiguedad. Es por ello que en los últimos años los algoritmos

de aprendizaje en general y de aprendizaje de reglas difusas en particular, han cobrado una creciente importancia. A medida que la extracción de conocimiento se hace cada vez más necesaria y compleja (dada la complejidad de los datos), los algoritmos de aprendizaje han necesitado evolucionar y adaptarse para seguir siendo eficientes.

El termino "eficiente", hace referencia a un conjunto de argumentos (parámetros) que varían entre sí. Puede hacerse referencia, por tanto, a un algoritmo eficiente en tiempo, en complejidad del conocimiento extraído, en capacidad de predicción, etc. Así, lo deseable sería encontrar un buen balance entre todas estas medidas, lo que supone uno de los principales problemas a resolver.

En concreto, las técnicas que se proponen se centran en el estudio de la precisión o capacidad de predicción de la base de reglas, el tiempo requerido para obtener el modelo y la interpretabilidad, esta última a nivel de base de reglas (número de reglas) y a nivel de regla individual (número de condiciones y aproximación al lenguaje natural de las mismas).

Mejorar la capacidad de predicción de la base de reglas enlaza directamente con la idea de considerar la relevancia indirecta de los atributos de entrada. Como hemos mencionado anteriormente, en ocasiones los datos, aunque abundantes, pueden no ser suficientemente representativos, es decir, aportan poca información a la hora de establecer un criterio claro de clasificación. Una de las técnicas emergentes en los últimos años y que permite trabajar con conjuntos de datos que presentan estas características, es la *construcción de atributos*. Esta técnica consiste básicamente en generar nuevos atributos a partir de la combinación de las variables originales. El aporte más importante de la construcción de atributos es la posibilidad de extraer información adicional, no evidente, además de la individual asociada a cada variable.

Esta idea de construir nuevos atributos se plantea por medio del uso de relaciones y funciones en el antecedente de reglas difusas. Ademas del previsible incremento de la capacidad de predicción, es importante mencionar que se espera una mejora en la interpretabilidad de las reglas, dado que deben aproximarse más al uso del lenguaje natural. Sin embargo, es posible que las principales deficiencias vengan de la mano de un incremento en la complejidad de la base de reglas y del tiempo empleado para obtener el modelo.

Frente a esta nueva situación, en la que el incremento de la complejidad de la base de reglas iría en detrimento de la interpretabilidad de la misma, conectamos con la segunda parte del objetivo en la que extendemos el SC strategy para que sea capaz de revisar el conocimiento extraído y decidir, en cada caso, si es conveniente sustituir alguna de las reglas previamente aprendidas. Este nuevo modelo permite reducir la complejidad de la base de reglas, incluso el numero de condiciones de las mismas, además del tiempo de ejecución. El inconveniente que presenta esta estrategia de revisión del

conocimiento es la dependencia en el proceso de aprendizaje de la aparición de reglas muy generales. Estas reglas, difíciles de reemplazar, conducen hacia el aprendizaje de reglas muy específicas que tienen poca incidencia en la representación de ejemplos. Es por esta razón, que conviene guiar el proceso de aprendizaje hacia un comportamiento en el que se retarde lo máximo posible la aparición de reglas muy generales y se limite la especificidad de las reglas en etapas finales. Esto lo conseguiremos acotando los espacios de búsqueda por medio de una condición de completitud, que permitirá establecer un umbral de cobertura mínimo para cada regla extraída.

Es importante destacar que este modelo de revisión del conocimiento, establece las bases para evolucionar hacia un nuevo modelo de aprendizaje incremental.

En las siguientes secciones desarrollaremos con detalle los métodos descritos anteriormente, haciendo especial hincapié en los cambios del modelo de recubrimiento secuencial de cada algoritmo.

1.7 Descripción de los Capítulos

Esta memoria está estructurada en 7 capítulos, incluyendo el actual, en el que se introducen las ideas básicas que enmarcan el contexto donde se ubica el trabajo llevado a cabo. De igual forma, se especifican las principales motivaciones y objetivos del mismo.

De esta forma, en el Capítulo 2 se aborda el problema de aprendizaje de sistemas basados en reglas difusas. Además, también se describe el modelo iterativo de aprendizaje de reglas, que constituye la base en la que se sustentan los métodos aquí propuestos. Finalmente, en el capítulo se describen algunos ejemplos representativos de métodos de aprendizaje basados en algoritmos genéticos.

El Capítulo 3, se centra en la descripción de **SLAVE**, un algoritmo de aprendizaje basado en reglas difusas caracterizado por el uso de un algoritmo genético dentro de un modelo iterativo. De igual manera, se estudian en profundidad dos de sus evoluciones más importantes: **SLAVE2** y **NSLV**. El capítulo finaliza con un exhaustivo estudio experimental donde se resumen los resultados más destacados.

El objetivo en el Capítulo 4 es introducir el uso de la técnica conocida como *construcción de atributos* bajo un enfoque de aprendizaje con genéticos. En este sentido, la construcción de atributos se ha llevado a cabo mediante dos estrategias distintas: el uso de relaciones en el antecedente de las reglas difusas y por medio del uso de funciones en el antecedente de dichas reglas. Como consecuencia, se presentan tres algoritmos: dos correspondientes a la aplicación de cada técnica por separado (**NSLV-R** y **NSLV-F**) y uno más combinando ambas técnicas en el mismo método de aprendizaje (**NSLV-FR**). A continuación, se detallan los resultados obtenidos por las

distintas propuestas junto con un estudio comparativo entre ellas. Finalmente, la última sección muestra una aplicación real de nuestro algoritmo de construcción de atributos (NSLV-FR) sobre un problema de imágenes obtenidas por satélite. Esta última sección es el resultado del trabajo desarrollado durante una estancia de tres meses en la Universidad Aristóteles de Tesalónica (Grecia), bajo la supervisión del profesor D. John B. Theocharis.

El uso de técnicas de construcción de atributos normalmente da como resultado bases de reglas más interpretables. Aquí el término "interpretable" hace referencia a una mayor aproximación al lenguaje natural. Sin embargo, la realidad es que estas técnicas suelen incrementar el espacio de búsqueda de soluciones, lo que se traduce en bases de reglas formadas por un mayor número de reglas, cada una de las cuales hace uso de un número mayor de condiciones. Este incremento tanto de reglas como del número de condiciones, es contrario a determinadas medidas de interpretabilidad. Debido a esto, en el Capítulo 5, se propone una modificación del modelo de aprendizaje iterativo de manera que el algoritmo incorpore la capacidad de revisar el conocimiento extraído en cada iteración. El capítulo incluye un estudio experimental en el que el método propuesto (**NSLV-AR**) demuestra un buen comportamiento a la hora de trabajar con una amplia variedad de problemas.

Con el propósito de aprovechar las principales virtudes de las estrategias utilizadas por las distintas propuestas expuestas con anterioridad, en el Capítulo 6 se define un modelo integrado capaz de combinar técnicas de construcción de atributos con revisión del conocimiento (**SLAVE3**). Además de esto, hay que ubicar en el capítulo tres secciones más como parte del estudio experimental global. En la primera de ellas, se incluye una comparación completa entre las principales propuestas presentadas en la tesis. La segunda sección está dedicada a comparar SLAVE3 con otros conocidos algoritmos de aprendizaje. Por último, en la tercera se establecen los principios básicos para la definición un modelo de aprendizaje incremental.

El Capítulo 7 contiene las conclusiones que se derivan de los resultados obtenidos por las distintas propuestas descritas en esta tesis, así como sus principales aportaciones. De igual manera, se aportan algunas consideraciones que podrían ser potencialmente interesantes en futuras implementaciones.

1.8 Publicaciones

Esta sección está dedicada a enumerar todas aquellas publicaciones en Congresos y Revistas que se han llevado a cabo durante el desarrollo de esta tesis doctoral.

- García, D., González, A., Pérez, R., "A two-step approach of feature construction for a genetic learning algorithm", *IEEE International*

Conference on Fuzzy Systems, 1255–1262 (2011, June).

- García, D., González, A., Pérez, R., "An iterative strategy for feature construction on a fuzzy rule-based learning algorithm", *11th International Conference on Intelligent Systems Design and Applications (ISDA)*, 1235–1240 (2011, November).

- García, D., González, A., Leyva, E., Pérez, R., "Un estudio experimental del uso de dominios con intensificaciones", *Congreso Español sobre Tecnologías y Lógica Fuzzy* (2012, February).

- García, D., González, A., Pérez, R., "A filter proposal for including feature construction in a genetic learning algorithm", *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, **20 (supp 02)**, 31–49 (2012, October).

- García, D., González A., Pérez R., "An empirical study about the behavior of a genetic learning algorithm on searching spaces pruned by a completeness condition", *Proceedings of the 2013 IEEE International Workshop on Genetic and Evolutionary Fuzzy Systems, GEFS 2013 - 2013 IEEE Symposium Series on Computational Intelligence, SSCI*, 8–15 (2013, April).

- García, D., González, A., Pérez, R., "A new iterative model to simplify the knowledge extracted on a fuzzy rule-based learning algorithm", *IEEE International Conference on Fuzzy Systems* (2013, July).

- García, D., González, A., Pérez, R., "Overview of the SLAVE learning algorithm: A review of its evolution and prospects", *International Journal of Computational Intelligence Systems*, **7 (6)**, 1194–1221 (2014, August).

- García, D., González, A., Pérez, R., "A feature construction approach for genetic iterative rule learning algorithm", *Journal of Computer and System Sciences*, **80(1)**, 101–117 (2014, December).

- García, D., Gámez, J. C., González, A., Pérez, R., "Using a sequential covering strategy for discovering fuzzy rules incrementally", *Proceedings of the IEEE International Conference on Fuzzy Systems* (2015, August).

Chapter 2

Learning fuzzy rule bases under the Iterative Rule Learning approach

2.1 Introduction

Generally, FRBSs can be considered as rule-based systems using *fuzzy logic* (FL) for representing knowledge related to problems of different kinds. As it is explained in [23], these types of systems are extensions of classic rule-based systems, because they work with *"IF-THEN"* rules in which both, antecedent and consequent use linguistic or fuzzy variables. They mainly present two advantages with regards to the first ones:

- they allow handling vague information, and
- the inference methods are more robust and flexible thanks to the use of FL.

So, according to the shape of the fuzzy rules and the types of inputs/outputs, three main classes of FRBSs can be distinguished:

- **Mamdani FRBSs**: Proposed by Mamdani [72], they are mainly characterized by four elements: a *KB*, an *inference system* and *Fuzzyfication and Defuzzyfication interfaces* [73]. Likewise, fuzzy rules are composed by input/output linguistic variables, which give a high level of

interpretability. These linguistic rules present the following structure:

$$IF\ X_1\ is\ A_1\ and\ ...\ and\ X_n\ is\ A_n\ THEN\ Y\ is\ B.$$

with X_i and Y input and output linguistic variables and A_i and B linguistic labels with fuzzy sets associated.

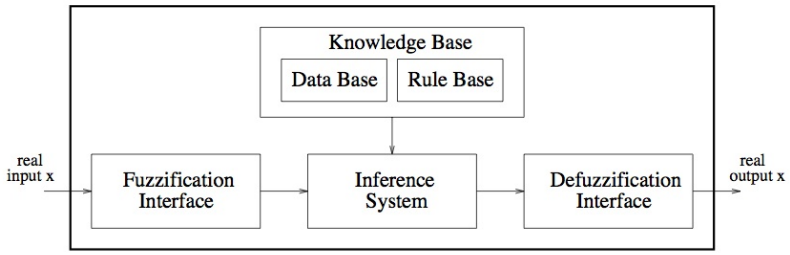


Figure 2.1: Structure of a Mamdani FRBS

- **Takagi-Sugeno-Kang FRBSs**: Proposed by Takagi, Sugeno and Kang [111, 109], it is a rule-based model in which the antecedent is formed by linguistic variables and the consequent is represented through a function applied over the input variables, frequently described by means of a lineal function. These kind of rules ara commonly described by:

$$IF\ X_1\ is\ A_1\ and\ ...\ and\ X_n\ is\ A_n\ THEN$$

$$Y = p_1 * X_1 + ... + p_n * X_n + p_0.$$

with X_i and Y input and output variables, A_i linguistic labels with fuzzy sets associated and p_i members of a vector of real parameters.

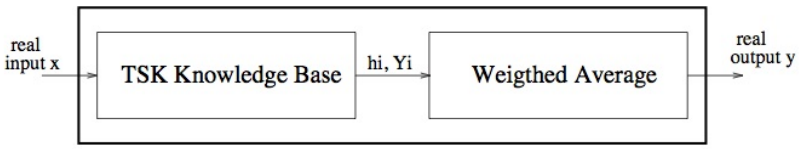


Figure 2.2: Structure of a TSK FRBS

In Figure 2.2, Y_i represents the individual output provided by the i-th rule and h_i refers to the matching degree between the antecedent of the i-th rule and the inputs to the system.

Also in [23], two main tasks are highlighted when designing a FRBS. One of them is related to the inference engine, that is, the operators used in the inference process while the other one is related to the way to obtain a knowledge base describing a specific problem. Both tasks directly affects the accuracy of the FRBS.

At the same time, the available information when designing a FRBS can be basically of two different types: numeric (observing the system) and linguistic (using information given by an expert). Thus, in order to obtain a KB representing the problem, two methods can be applied:

- Expert knowledge: the linguistic labels associated to each variable are defined by an expert in the domain.
- Learning methods based on numeric information: through numeric inductive learning methods.

Working with linguistic FRBSs helps improving the interpretability of rules and, as a consequence, of the model obtained. This makes the results easily understandable by humans.

2.2 Learning FRBSs

In the literature it is possible to find several examples of fuzzy rule-based learning algorithms, which can be classified according to the technique they use for learning in the following groups:

- *Ad hoc*: considered those which are based in the learning of fuzzy rules guided by a covering criteria of data belonging to the example set. According to [17], these methods present some interesting advantages:
  - Due to their simplicity, they are easily understandable and interpretable.
  - Short time-consuming when learning non high-dimensional problems.
  - They are suitable for obtaining preliminary fuzzy models in the modeling process.

  Some of the most representative works are those published by Wang and Mendel [116] and Nozaki et al. [88].
- *Neural Networks (NNs)*: focused in the shaping of the biological functions of the human brain, according to [114]. From a technical point of view, artificial neural networks (ANNs) are defined as computational models that emerged as mathematical formalization of the structure

of the brain [4]. Many other definitions can be found in the literature about this topic [54, 68, 84, 102, 86], but in the first work Vieira et al. establish a relation among neuro fuzzy systems and NNs. First of all, some advantages and disadvantages of NNs are noted.

Some of the advantages are:

- The learning capacity.
- The generalization capacity.
- The robustness against noise.

On the other hand, the main disadvantages are:

- It is difficult to interpret the functionality.
- It is difficult to determine the number of neurons and layers

From these advantages and disadvantages it is suggested the combination of NNs and fuzzy systems in order to increase the positive aspects of both approaches and reduce the negative ones. Thus, the combination of techniques involving neural networks and fuzzy logic is known as neuro fuzzy systems.

Some examples of algorithms using NNs for classification problems are [63, 61, 76].

- *Genetic Algorithms*: which are general-purpose search algorithms that use principles inspired by natural population genetics to evolve solutions to problems [4, 41, 55]. The basic idea is to maintain a chromosome population representing potentially useful solutions for a specific problem [92]. These chromosomes evolve along time through a process of competition and controlled variation. They also have a goodness measure (*fitness function*), which represents the adaptation to the solution they represent. The new individuals are created through genetic operators such as crossover and mutation operators. GAs have been successfully applied on searching and optimization problems. They are specially useful due to their ability to exploit the stored information about the searching space. Thus, it is possible to guide future searchings to better subspaces.

  A GA starts with a population of randomly generated chromosomes, and obtains better chromosomes by applying genetic operators, modeled on the genetic processes occurring in nature. During each iteration an evaluation function (*fitness function*) is used to distinguish between good and bad solutions. The process of going from the current population to the next one constitutes one iteration in the execution of a genetic algorithm.

Although there are many possible variants of GAs, the main mechanisms guiding their behavior imply changing in each iteration the population according to the three operations below:

- Evaluation of the individual of the population.
- Creation of an intermediate population.
- Creation of a new population using the genetic operators.

These steps are repeated until no improvement is achieved or after a maximum number of iterations (established by the user) is reached. Figure 2.3 shows the main structure of a GA [22].

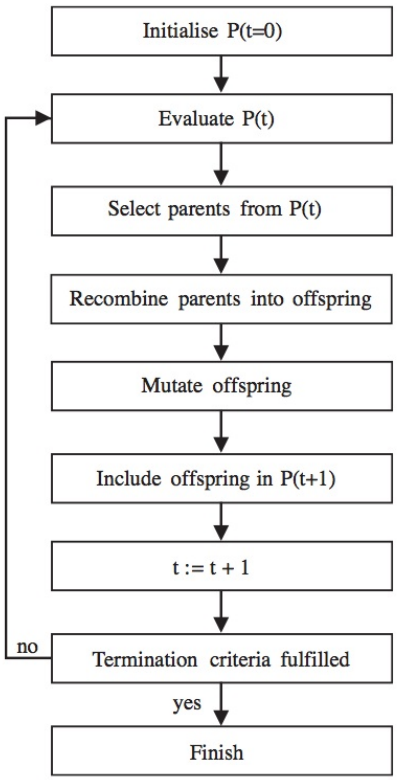


Figure 2.3: General structure of a GA. Figure extracted from [22].

In the last years the importance of the application of GAs in the learning problems has grown. As it was previously said, the GAs give an evolutionary nature to the searching of solutions for a problem. The main disadvantage is to find an uniform representation of the information of the problem as well as the solutions.

Considering this idea and as explained in [42] and [23], it is possible to distinguish among three classic approaches of genetic learning:

– **The Michigan approach** [56, 12]: The chromosomes are individual rules and a rule set is represented by the entire population. The collection of rules are modified over time via interaction with the environment. This model maintains the population of classifiers with credit assignment, rule discovery and genetic operations applied at the level of the individual rule. Figure 2.4 shows the learning process under the Michigan approach [22].

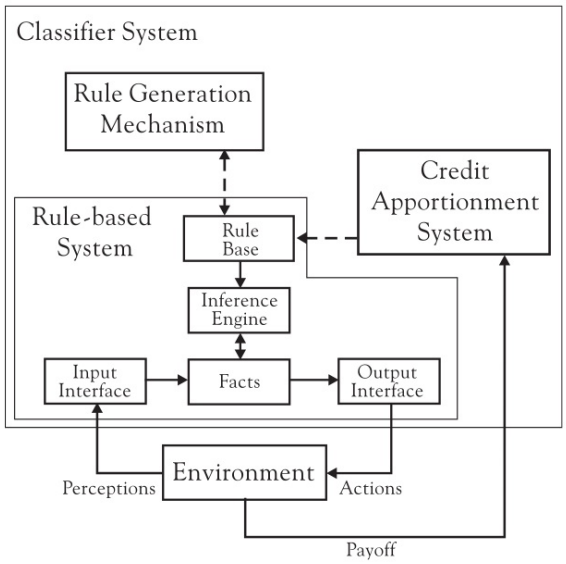


Figure 2.4: Michigan approach. Figure extracted from [22].

– **The Pittsburgh approach** [104]: Each chromosome encodes a whole classifier set. Credit is assigned to the complete set of rules via interaction with the environment. Crossover serves to provide a new combination of rules and mutation provides new rules. In some cases, variable-length classifier sets are used, employing modified genetic operators for dealing with these variable-length and position independent genomes. Figure 2.5 shows the learning process under the Pittsburgh approach [22].

– **The iterative rule learning (IRL) approach**: A chromosome codes an individual rule and a novel rule is adapted and added to the rule set in a iterative fashion in every GA run. The GA provides a partial solution to the learning problem and, contrary to both previous approaches, it is run several times to obtain the

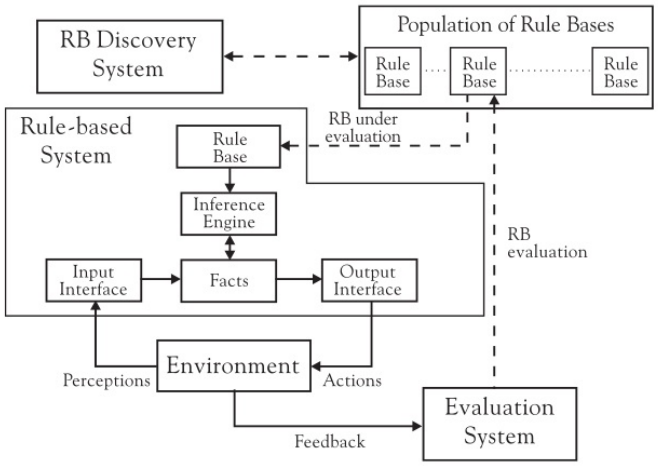


Figure 2.5: Pittsburgh approach. Figure extracted from [22].

complete set of rules. This approach was initially used in SLAVE [45, 37] and SIA [113] systems.

As it is explained in [16], the role of the GA in the Pittsburgh and Michigan models is different depending on the level in which it is applied. Because of that, both approaches suffer some problems.

The main problem in the Michigan approach is related to the conflict between the individual and collective interests of system classificators. In this model, the rules cooperate in order to receive a gratification but they also compete with each other under the GA action.

In the Pittsburgh model the behavior is different, because the reproductive competition is carried out at a ruleset level instead of a single rule. The inconvenience is that the maintenance and evaluation of these rulesets needs a higher computational cost (memory and processing time).

In the iterative approach, contrary to the Michigan model, only the best individual is considered as solution, discarding the rest of chromosomes of the population.

Anyway, as the methods presented in this dissertation are based on the IRL approach, next section will briefly describe some of its most important points.

2.3 The IRL approach

This approach was initially created to combine the best characteristics of the Michigan and Pittsburgh approaches and to solve their main problems. In this model, the GA gives a partial solution to the learning problem, reducing the searching space of possible solutions. With the idea of optimizing this process, the GA is included in an iterative structure (that was previously referred as SC strategy [81]), which is the basic scheme of the approach and responds to the following description:

1. Use a GA to obtain a rule for the system.
2. Incorporate the rule into the final set of rules.
3. Penalize this rule.
4. If the set of rules obtained is adequate to represent the examples in the training set, the system ends up returning the set of rules as the solution. Otherwise, return to step 1.

An easy way to penalize the rules already obtained, and going on learning new rules, consist in removing from the training set those examples that are covered by the set of rules.

So, in order to implement a learning algorithm based on GAs using the IRL approach, we need at least the following:

1. A criterion for selecting the best rule in each iteration.
2. A penalization criterion.
3. A criterion for determining when are available enough rules to represent the examples in the training set.

The first criterion is normally associated with one or several characteristics to measure the goodness of a rule like the consistency or the simplicity, for example. The second criterion is often associated with the elimination of the examples covered by the previous rules. Finally, the third one is related to the completeness of the set of rules and must be taken into account when the examples in the training set are sufficiently covered, so that no more rules are needed to represent them.

2.4 Genetic algorithms based methods

This section tries to collect some representative genetic fuzzy rule learning algorithms following the IRL approach. A brief description of each algo-

rithm is given using as a reference the classification appearing in the KEEL platform [3]. Some of them have been used in the experimental study.

2.4.1 SLAVE

As it was said in Chapter 1, SLAVE was the first fuzzy rule-based learning algorithm using the IRL approach. It iteratively extracts a single rule, using a GA. Then, this rule is added to the rule base and the process is repeated until no more rules improving the rule base are found or a maximum number of iterations without changes is reached.

It is characterized by the model of rule used and by the rule selection heuristic criterion based on the consistency and completeness conditions.

A detailed description of SLAVE and its evolutions NSLV and SLAVE2 is given in Chapter 3.

2.4.2 MOGUL

As its name suggests, MOGUL (a Methodology to Obtain Genetic fuzzy rule-based systems Under the iterative rule Learning approach) [21, 20], works under the IRL approach and it may be used to generate different types of fuzzy rule-based systems (FRBSs): approximate Mamdani-type [73], and TSK-type ones [111].

In order to improve the accuracy of the FRBS, MOGUL handles two different tasks, the simplification of the knowledge base and the refinement of the fuzzy rules by adjusting their membership functions.

With this purpose, the postprocessing stage is divided in two others: the genetic multisimplification process and the evolutionary tuning process. The first one is capable of generating different definitions that present the best possible cooperation between the fuzzy rules and thereby the best possible behaviour. Then, the evolutionary tuning process is applied over these definitions and the most accurate is the definition given as the output of the multistage GFRBS. Therefore, a knowledge base that does not present the best behaviour after the second stage may be the best after the third stage due to the fact that the new membership function shapes make its rules cooperative in a better way.

Some other important aspects of MOGUL to take into account are:

- The designer is allowed to build the generation stage by using different kinds of algorithms, rather than only a GA. It is possible to employ a non evolutionary inductive algorithm or an evolution strategy [10], instead of the usual GA. The operation mode is the same, but the difference is the speed of the generation process, which is higher in the former case.

- Several important statistical properties have to be verified by the knowledge base in order to obtain an FRBS that presents good behaviour [44]. The satisfaction of completeness and consistency is considered in the GFRBSs obtained from MOGUL in order to improve the behaviour of the generated knowledge bases.
- Focusing on the evolutionary algorithm (EA) search, there is a need to make use of suitable techniques to develop an accurate trek on the search spaces tackled in each stage to obtain the best possible solutions. Several factors have to be considered to reduce the search space complexity and to perform an adequate exploration and exploitation over it to allow the search process to be effective. MOGUL proposes the use of many techniques.
- The available knowledge is incorporated into the genetic learning process in order to improve its performance by either directly incorporating partial definitions obtained from expert knowledge or using the available knowledge to generate de initial population of the evolutionary algorithms considered in each stage.

2.4.3 Logitboost

Logitboost [29, 90], is a boosting method that tries to minimize the likelihood of the classifier, which in turn is restricted to a parametric family of density functions. It is used for learning fuzzy rule based classifiers, using the following structure:

The algorithm produces rules with a single consequent in two-class problems, and rules with more than one consequent in multiclass problems. The search is carried out through a genetic algorithm, that finds the combination of antecedents (the fuzzy set A) which, in combination with its optimal value of s^j best approximates the residual, in the weighted least squares sense.

2.4.4 GCCL

Ishibuchi et al. present in [62] a fuzzy genetics-based machine learning method for multidimensional pattern classification problems with continuous attributes.

The proposed method uses genetic operators such as selection, crossover and mutation for generating combinations of antecedents of fuzzy if-then rules. The outline is shown below:

- **Step 1**: Generate an initial population of fuzzy if-then rules.
- **Step 2**: Evaluate each fuzzy rule in the current population.
- **Step 3**: Generate new rules through the genetic operations.

$f_{i0k} = 0$
For step number $j = 1, \ldots, N$
For class number $k = 1, \ldots, p$
for $i = 1, \ldots, n$ do $p_{ijk} = e^{f_{ij-1k}}/(1 + e^{f_{ij-1k}})$
for $i = 1, \ldots, n$ do $w_{ijk} = p_{ijk}(1 - p_{ijk})$.
Find A^j that minimizes

$$\text{fitness}(A^j) = \sum_i^n w_{ijk} \left(s^j \cdot A^j(x_i) - \frac{y_{ik} - p_{ijk}}{w_{ijk}} \right)^2$$

$$\text{where } s^j = \frac{\sum_i (y_{ik} - p_{ijk}) A^j(x_i)}{\sum_i w_{ijk} [A^j(x_i)]^2}$$

for $i = 1, \ldots, n$ do $f_{ijk} = f_{ij-1k} + s^j \cdot A^j(x_i)$
if $s^j > 0$ then Emit the Rule "**if x is** A^j **then** $\mathbf{t}(c_k)$=s^j"
else Emit the Rule "**if x is** A^j **then** $\mathbf{t}(c_1)$=-s^j ... $\mathbf{t}(c_k)$=**0** ... $\mathbf{t}(c_p)$=-s^j"
End for
End for

Figure 2.6: Outline of the basic version of backfitting applied to a logistic extended additive model or Logitboost. The scheme has been extracted from [90].

- **Step 4**: Replace a part of the current population with the new generated rules.
- **Step 5**: Terminate the algorithm if a stopping condition is satisfied, otherwise return to Step 2.

According to the results given in the paper, the proposal achieves both high performance and high comprehensibility and demonstrates to work well for real-world pattern classification problems with more than ten continuous attributes.

2.4.5 SGERD

In [74], Mansoori et al. propose a steady-state genetic algorithm to extract a compact set of fuzzy rules from numerical data, called SGERD. It is generational and population-based and its generations are finite and bounded to the problem dimension. Regarding to the selection process only the best individuals can survive. Each parent produces a finite number of offspring through reproduction. The rule selection mechanism in SGERD induces competition among rules by only considering the performance of each rule. The cooperation among rules is carried out through a heuristic approach

that selects only the most cooperative rules among the final population for a rule base.

The algorithm is described through the following steps:

Inputs: m labeled patterns of an n-dimensional M-class problem and a prespecified number of Q rules per class.

Outputs: Possibly $R = M \times Q$ fuzzy classification rules.

- **Step 1**: i=1 (i: generationnumber).
- **Step 2**: Generate all fuzzy rules having only one active antecedent variable (at most $C = 14 \times$ n candidate rules would be generated).
- **Step 3**: Determine the consequent class of each candidate rule using a measure for evaluating the confidence of a fuzzy association rule.
- **Step 4**: Divide the candidate rules into M groups according to their consequent class.
- **Step 5**: Rank, in descending order of their fitness values, the candidate rules in each group.
- **Step 6**: Choose the best Q rules from each class (i.e., possibly $R = M \times Q$ rules in total) as the population in the *i-th* generation. In the first generation only, choose the second best R rules as the auxiliary population and put away for mutation.
- **Step 7**: Increment i, **if** $i > n$, **goto Step 11**.
- **Step 8**: Use all individuals in the previous generation (i.e., R rules) as parents and do reproduction (i.e., crossover, mutation, or elitism) on them. That is, for each parent rule, generate as offspring all fuzzy rules having one more active antecedent variable than its parent, provided each new offspring be fitter than its parent. In this case, the number of offspring will totally be $R \times 14$ at most.
- **Step 9**: If no offspring fitter than the parents is produced in **Step 8**, **goto Step 11**.
- **Step 10**: Consider both parents and offspring in **Step 8** as candidate rules (at most $C = R + R \times 14$ rules in total) and **goto Step 3**.
- **Step 11**: Use $R = M \times Q$ rules (obtained in **Step 6** for the *i-th* generation) as the final population and stop. The actual length of these rules is i or less.

The authors indicate that SGERD demonstrate to be fast enough to be applied to high-dimensional data set including numerical attributes. They also list some of its main advantages:

- It is a simple an intuitive algorithm.
- It generates a few general rules that are short, accurate and interpretable.
- Its population size and number of generations are small, so it saves memory space.
- It is fast and can be applied to high-dimensional problems.

As it was said before, these are only a representative selection of some algorithms capable to obtain FRBSs using GAs. From now on, this dissertation will be focused in the description of algorithms working under the IRL approach.

Thus, next chapter will present the learning algorithm SLAVE and its evolutions, one of which, NSLV, has been used as the base algorithm for the development of the later methods.

Chapter 3

The learning algorithm of SLAVE and its evolutions: SLAVE2 and NSLV

3.1 SLAVE

3.1.1 Motivation

In the previous chapter we placed and introduced SLAVE as a fuzzy rule learning algorithm using genetic algorithms for learning single rules. Now, we will deeply describe this algorithm together with some of its most relevant evolutions.

SLAVE (Structural Learning Algorithm in a Vague Environment) was first proposed in 1994[51] and later developed in 1996[43] with the goal of extracting a set a fuzzy rules to represent a problem. At that time there were not many algorithms for fuzzy rule learning available. The only relevant proposals were those of Wang and Mendel[116] and Jang[63], published two years and one year earlier respectively. Both algorithms were focused mainly on control rule learning (regression problems), where fuzzy modeling had proven to be useful, so they were not designed specifically for classification problems. Although these proposals used different methodologies, they had in common the Mamdani rule model. The main drawback of these proposals was the limitation in the number of variables they could handle. When working with a high enough number of variables, the response time becomes

unmanageable.

Probably the origin of SLAVE lies to a large extent on the weaknesses of these proposals and the lack of a fuzzy rule learning proposal for classification problems. When SLAVE was developed, the main objective was to build a fuzzy rule learning algorithm targeting classification problems, featuring a performance similar to that provided by the classical learning algorithms, and able to manage problems involving many variables, missing data, etc.

3.1.2 What is SLAVE?

SLAVE is a fuzzy rule learning algorithm based on the use of a sequential covering strategy [81]. A prototypical description of this family of algorithms is shown below:

SEQUENTIAL-COVERING (Y,X,E,Learned-rules)

- Learned-rules ← {}
- Rule ← LEARN-ONE-RULE (Y, X, E)
- while PERFORMANCE (Rule,E) $>$ 0, do
  - Learned-rules ← Learned-rules + Rule
  - E ← E-examples correctly classified by Rule
  - Rule ← LEARN-ONE-RULE (Y, X, E)
- Learned-rules ← sort Learned-rules according to PERFORMANCE (Rule,E) over Examples
- return Learned-rules

where Y is the target attribute, X is the attribute set, E is the set of examples and *Learned-rules* is the output of the procedure containing the final set of rules. PERFORMANCE is a procedure that measures the contribution caused by the inclusion of the last rule in the rule base. It measures the increase in the degree of completeness that causes the last learned rule over the set of examples E.

To describe the implementation of this strategy in SLAVE, we must first define the type of rule and the process used to implement the LEARN-ONE-RULE procedure, together with other details that we mention below.

3.1.3 Description of the main elements

3.1.3.1 DNF model of rule

A very well known extension of a simple rule is the DNF rule, in which each input variable takes as possible values a set of linguistic terms whose

members are joined by a disjunctive operator. On the other hand, the output variable uses a common linguistic variable with a single associated value. SLAVE employs a fuzzy extension of the DNF rule model:

IF X_1 is A_1 and X_2 is A_2 and $\ldots$ and X_n is A_n

THEN Y is B with **weight w**

where $X_1, \ldots, X_n$ are the attributes, $A = (A_1, \ldots, A_n)$ are the values taken for each attribute, each A_i is a subset of D_i, the fuzzy domain of X_i, Y is the consequent variable and B is the value of the consequent variable. We denote this rule as $\boldsymbol{R_B(A)}$. Finally, w is a measure of the weight associated to the rule.

This rule model has been used in SLAVE to learn the structure of a rule, since when a variable takes all possible fuzzy values of its domain, it is not relevant for the consequent and therefore can be removed from the rule. Moreover, the DNF fuzzy rule model allows compacting the set of rules and makes it more interpretable.

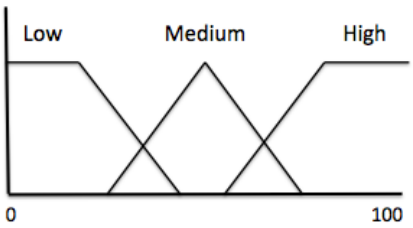


Figure 3.1: Fuzzy domains for variables X_1 and X_2.

A specific example of a DNF fuzzy rule is:

IF X_1 is {Low or Medium or High} and X_2 is {Medium or High}

THEN Y is 2 with **weight w**

where the fuzzy domains of X_1 and X_2 are described in Figure 3.1, and Y takes its values in a discrete domain with class in the set {1, 2, 3}. Since X_1 takes all the possible values of its domain, the previous rule is equivalent to:

IF X_2 is {Medium or High} **THEN** Y is 2 with **weight w**

Moreover, the label {Medium or High} can be interpreted as the convex hull of both labels [43, 45].

3.1.3.2 The LEARN-ONE-RULE function

SLAVE uses a genetic algorithm (**GA**) to implement the LEARN-ONE-RULE function. The input of this **GA** is a target attribute, representing the consequent variable, the complete set of antecedent variables and the set of examples, and the output is a single rule. In SLAVE, function LEARN-ONE-RULE is called following a particular order. It sets a specific value of the consequent class and this procedure iterates until all the rules needed for describing this class are obtained; then another class is selected and the process is repeated.

One of the main components of the LEARN-ONE-RULE is the criterion to extract a single rule. The main idea is to extract the rule that offers the best representation of the set of examples. The best rule criterion is related to an extension of the classical conditions of completeness and consistency. In order to propose a fuzzy version of these concepts we first need a way to decide whether an example is positive or negative for a rule, and a way to count the number of examples in both cases.

3.1.3.2.1 Positive and negative examples

The way to calculate the number of positive and negative examples for a rule $R_B(A)$ in SLAVE changed over the years. The first approximation for calculating the number of positive and negative examples[51] was based on a possibility measure. Thus, let a and b be two fuzzy sets in a common referential set U, and $*$ a t-norm. The compatibility between a and b was defined by the function:

$$\sigma(a,b) = sup_{x \in U}\{\mu_a(x) * \mu_b(x)\} \tag{3.1}$$

where $\mu_r(s)$ is the degree of membership of value s to the fuzzy set r.

In order to calculate the compatibility between two sets of fuzzy sets, let us consider Dom_1 and Dom_2, two domains consisting of fuzzy sets in a common referential set U, and $C_1 \subseteq Dom_1$ and $C_2 \subseteq Dom_2$, two sets of fuzzy sets. In this case, the compatibility between both sets follows this formula:

$$\sigma(C_1, C_2) = sup_{a \in C_1} sup_{b \in C_2} \sigma(a,b). \tag{3.2}$$

Considering these expressions it was possible to define the following measure of possibility:

$$Poss(A_i|e_i) = \frac{\sigma(e_i, A_i)}{\sigma(e_i, D_i)} \tag{3.3}$$

representing the adaptation between the i-th component of the example and the fuzzy set A_i, where $E = (e_1, e_2, \ldots, e_n)$ is an example, $A_i \subseteq D_i$ is

the value of variable X_i, and D_i is the domain of this variable. This concept is critical to the interpretation of two adjacent fuzzy sets as the convex hull of both.

From these definitions we could obtain two adaptation concepts, one for the antecedent part and another for the consequent part:

- Adaptation between the example and the antecedent of $R_B(A)$:

$$U(e, A) = *_{i=1...n} Poss(A_i|e_i). \tag{3.4}$$

- Adaptation between the example and the consequent of $R_B(A)$:

$$U(e, B) = \frac{\sigma(class(e), B)}{\sigma(class(e), F)} \tag{3.5}$$

where e is an example, $class(e)$ represents the consequent of the example, and * is a t-norm.

Using the previous concepts we can define the set of positive and negative examples for rule $R_B(A)$:

$$E^+(R_B(A)) = \{(e, U(e, A) * U(e, B))|e \in E\} \tag{3.6}$$

$$E^-(R_B(A)) = \{(e, U(e, A) * U(e, \overline{B}))|e \in E\}. \tag{3.7}$$

where $\overline{B}$ is the set of all the fuzzy values of D_i, except B. Finally, the number of positive examples and the number of negative examples for fuzzy rule $R_B(A)$ are:

$$n_E^+(R_B(A)) = |E^+(R_B(A))| \tag{3.8}$$

and

$$n_E^-(R_B(A)) = |E^-(R_B(A))| \tag{3.9}$$

respectively, where $|.|$ is the cardinality of a fuzzy subset.

3.1.3.2.2 Completeness and consistency

Completeness and consistency are two conditions typically used to extract rules. The completeness condition states that every example of some class must satisfy some rule from this class. On the other hand, the consistency condition states that if an example satisfies a description of some class, then it cannot be a member of a training set of any other class. Both conditions provide the logical foundation of algorithms for concept learning from examples.

From the classical definitions of completeness and consistency[78], the following two extensions were defined in[45]. Given an example set E, the degree of completeness of a rule $R_B(A)$ is:

$$\Lambda(R_B(A), E) = \frac{n_E^+(R_B(A))}{n_{E_B}} \tag{3.10}$$

where

$$n_{E_B} = \sum_{e \in E} U(e, B) \tag{3.11}$$

is the number of examples of class B in the training set.

In relation to consistency, we propose a soft extension of this measure by allowing some noise in the rules. Thus, to define the soft consistency degree we use the following set:

$$\triangle_E^k = \{R_B(A) | n_E^-(R_B(A)) < k n_E^+(R_B(A))\} \tag{3.12}$$

with

$$\triangle_E^0 = \{R_B(A) | n_E^-(R_B(A)) = 0\} \tag{3.13}$$

which represents the set of rules having a number of negative examples strictly less than a percentage (that depends on k) of the positive examples, and being

$$\triangle = P(D_1) \times P(D_2) \times \ldots \times P(D_n) \times F \tag{3.14}$$

where $P(\chi)$ denotes the set of subsets of χ, D_i is the domain of antecedent variables, and F is the domain of the consequent variable.

Thus, the degree to which a rule $R = R_B(A)$ satisfies the soft consistency condition is:

$$\Gamma_{k_1 k_2}(R, E) = \begin{cases} 1 & \text{if } R \in \triangle_E^{k_1} \\ \frac{k_2 n_E^+(R) - n_E^-(R)}{n_E^+(R)(k_2 - k_1)} & \text{if } R \in (\triangle_E^{k_2} - \triangle_E^{k_1}) \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

where $k_1, k_2 \in [0, 1]$ and $k_1 < k_2$, and $n_E^-(R)$, $n_E^+(R)$ are the number of positive and negative examples for rule R, respectively.

This definition uses two parameters: k_1 is the lower bound of the noise threshold, and k_2 is the upper bound. The above formula gives degree 1 to rules in $\triangle_E^{k_1}$, that is, rules having an admissible number of negative examples (measured as a percentage in k_1 of the number of positive examples). It gives degree 0 to rules out of $\triangle_E^{k_2}$, that is, rules having an excessive number of negative examples (measured as a percentage in k_2 of the number of positive

examples). If $k_1 < k_2$ then $\triangle_E^{k_1} \subseteq \triangle_E^{k_2}$, so a linear variation is assigned to rules between both extremes.

After an experimental study, and from the publication of [48], the values for k_1 and k_2 were set to $k_1 = 0$ and $k_2 = 1$. In this case, the above expression is equivalent to:

$$\Gamma_{0,1}(R, E) = \frac{n_E^+(R) - n_E^-(R)}{n_E^+(R)}. \tag{3.16}$$

The criterion to select the best rule given a set of examples in SLAVE is finally

$$\Lambda(R, E) \times \Gamma_{0,1}(R, E) = \frac{n_E^+(R) - n_E^-(R)}{n_{E_B}}. \tag{3.17}$$

3.1.3.2.3 λ-covering concept

The adaptation between an example and a rule should not be a crisp concept when working with fuzzy information. In this way, an example has a degree of adaptation to a fuzzy rule. An important task is to know the minimum degree of adaptation that should exist between an example and a rule for considering that the example satisfies the rule. In order to solve this problem, SLAVE uses a parameter called $\lambda \in [0, 1]$. The meaning of λ is exactly the minimum adaptation required to consider that the example is covered by the rule. λ is an input parameter of the learning algorithm, and it plays an important role in the design of the algorithm. Low values of λ imply a low requirement for adaptation between examples and rules, and this fact has two consequences: The first one is that the system searches a small number of very general rules, and the second one is that there is probably too much overlap among rules competing for classifying, which often leads to less predictability. On the other hand, values of λ very close to 1 tend to force a very "crisp" rule behavior in relation to examples, which results in a greater number of more specific rules.

The concept of λ-covering is used for the removal of examples correctly classified in the sequential covering algorithm and in the termination condition.

From the work published in [51] and after an experimental study, the value of this parameter was set to $\lambda = 0.8$.

3.1.3.3 The genetic algorithm used in SLAVE

SLAVE, using the above criterion for selecting the best rule, employs a genetic algorithm to implement the LEARN-ONE-RULE function. Next we will describe the main components that define the behavior of the genetic algorithm.

3.1.3.3.1 Representation of a population element

Related to the genetic representation, SLAVE uses a binary codification. If the database has n antecedent variables

$$X_1, \ldots, X_n$$

each having an associated fuzzy domain D_i with m_i components, the antecedent of a rule is any element of

$$P(D_1) \times \ldots \times P(D_n),$$

and it is encoded as a vector of $m_1 + \ldots + m_n$ zero-one components. The value of each component $(m_1 + \ldots + m_{r-1} + s)$ is 1 if the s-th element in domain D_r is a value of variable X_r, and 0 otherwise.

In order to better understand the representation, let us consider the following example:

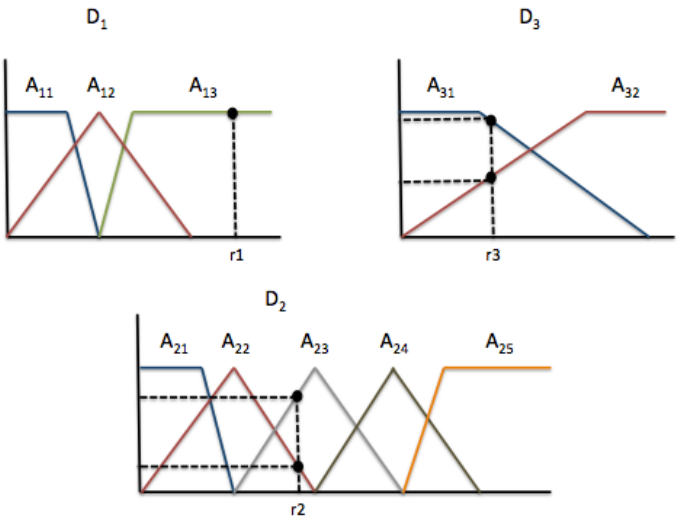


Figure 3.2: Domains of variables X_1, X_2 and X_3.

Example 1 *Let us assume that we have three variables X_1, X_2, and X_3, such that their associated fuzzy domains are*

$$D_1 = \{A_{11}, A_{12}, A_{13}\}$$

$$D_2 = \{A_{21}, A_{22}, A_{23}, A_{24}, A_{25}\}$$

$$D_3 = \{A_{31}, A_{32}\}.$$

These domains are represented in Figure 3.2. In this case, a vector 1100010111 represents the following antecedent:

$$X_1 \text{ is } \{A_{11}, A_{12}\},\ X_2 \text{ is } \{A_{23}, A_{25}\} \text{ and } X_3 \text{ is } \{A_{31}, A_{32}\}.$$

Since X_3 takes all possible values from domain D_3, the antecedent is equivalent to:

$$X_1 \text{ is } \{A_{11}, A_{12}\} \text{ and } X_2 \text{ is } \{A_{23}, A_{25}\}.$$

3.1.3.3.2 Generation of the initial population

The generation of the initial population is an important aspect in the process of getting antecedents with a high possibility of guiding the search process toward good solutions. The procedure used in SLAVE consists in randomly taking a subset of examples among those with the current consequent that have not been eliminated yet. For each of these examples, the most specific antecedent with the highest adaptation to the example is selected.

The procedure for selecting the initial population is similar to the one used in AQ algorithms[79], as the generation of the initial antecedent for a class uses examples of this class in the training set as a starting point, and the genetic process can be considered as a generalization process over the chosen antecedents.

Example 2 *Let X_1, X_2, and X_3 be variables with the associated domain shown in Figure 3.2, and let (r1, r2, r3) be the randomly selected example from the training set for a class. The most specific antecedent that best represents it would be:*

$$X_1 \text{ is } A_{13} \text{ and } X_2 \text{ is } A_{23} \text{ and } X_3 \text{ is } A_{31}$$

with the binary representation 0010010010.

3.1.3.3.3 Evaluation function

The main purpose of the function LEARN-ONE-RULE is to find the best rule verifying the completeness and consistency conditions to the highest possible degree, and for this task we use a genetic algorithm. According to that, given a rule R and a set of training examples E, we define the fitness function of the genetic algorithm in the following way:

$$fitness(R, E) = \Lambda(R, E) \times \Gamma_{0,1}(R, E) \tag{3.18}$$

where $\Gamma_{0,1}(R, E)$ is the degree to which the rule R satisfies the soft consistency condition and $\Lambda(R, E)$ is its degree of completeness, previously defined.

The combination of these two values provides rules that are simultaneously complete and consistent to the highest degree. Furthermore, we use the product t-norm, as the combination through this operator is very interactive compared to other proposals.

3.1.3.3.4 Genetic operators

The following genetic operators are used in SLAVE to generate new populations:

- **Selection operator**

  This is a selection model that sorts the elements in the population using its fitness valuation and assigns a probability selection to each position in the population.

- **Crossover operator over two points**

  This type of crossover establishes two cutoff points between two elements in the population and exchanges the central segment, like the one shown in Figure 3.3.

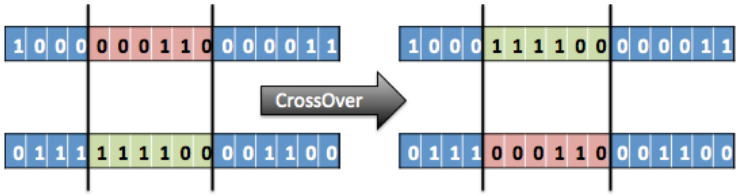


Figure 3.3: Crossover operator over two points.

- **Mutation operator**

  In the genetic algorithm used in SLAVE, this operator changes one gene of an element in the population with a certain probability.

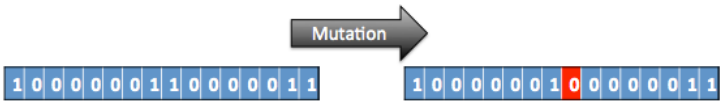


Figure 3.4: Mutation operator.

- **Generalization operator**

  This operator tries to clarify the rules returned by the learning algorithm and make them more understandable, and only acts over those

variables that have an associated ordered domain. We say that an antecedent variable with an associated ordered domain is stable if there is a single continuous sequence of 1-values in the binary representation of its value. We say that a variable is unstable if there are several continuous sequences of 1-values in the binary representation of its value.

Taking all this into account, the generalization operator tries to obtain stable variables by removing their unstable regions. The behavior is shown below (Figure 3.5).

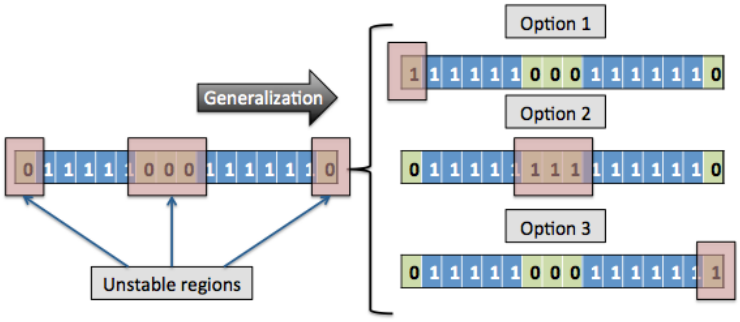


Figure 3.5: Generalization operator.

3.1.3.3.5 Termination condition

The implementation of this condition is necessary to distinguish between a class that has at least one rule and a class that does not. The idea is to make a more exhaustive search when we want to find the first rule of a class, and relax this search process when we already have some rules for a class.

The genetic algorithm returns the best rule for the last population if one of the following conditions is verified:

- The number of iterations is greater than a fixed limit.
- The fitness function of the best rule in the population does not increase the value during at least a fixed number of iterations and there are some other rules for the class we are working with.
- No rules with this value of the consequent have been obtained before, but the fitness function does not increase the value during a fixed number of iterations and the current best rule λ-covers at least one example from the training set.

3.1.3.4 Other components of SLAVE

3.1.3.4.1 The role of PERFORMANCE

The sequential covering previously described uses a *PERFORMANCE (Rule,E)* function, which receives a rule (*Rule*) and the set of examples (E) and measures how representative is that rule over the examples. In SLAVE this function is related to the concept of completeness. Specifically, when a new rule is added to the set of learned rules, the completeness of the set increases. When there is a non-zero increase, the new rule is considered useful and relevant. The rule is then added to the set of learned rules and the algorithm starts a new iteration. On the other hand, if the PERFORMANCE function is equal to zero, the rule is considered irrelevant, so it is discarded and the iterative process ends (as the rule base can not be improved anymore).

3.1.3.4.2 Removing examples

Another important aspect of the sequential covering is the removal of examples correctly classified by the learned rule. In each iteration, SLAVE removes the examples that were λ-covered by the last learned rule (i.e., the examples considered as sufficiently well classified).

3.1.3.4.3 Inference Model

SLAVE uses a typical fuzzy inference mechanism for classification: The winner rule. This model has a simple description.

Let $Rules = \{R_{B_1}(A_1), \ldots, R_{B_q}(A_q)\}$ be the set of rules and e an example. The inference engine assigns to the examples the class B_j related to the associated rule $R_{B_j}(A_j)$ verifying

$$U(e, A_j) \geq \max_{0 \leq i \leq q} \{U(e, A_i)\}. \tag{3.19}$$

When working with rules, it may occur that two or more rules describing different concepts can be applied to the same input system. In this case, it is necessary to establish a way to decide which of the possible rules should be applied.

The conflict resolution mechanism used in this version is based on rules confidence. This confidence degree is calculated taking into account the behavior of each rule over the training set, and it is related to their prediction capability. Thus, we define the weight of a rule as a value in [0, 1] that measures the relation between the examples correctly represented over the training set and the total set of examples covered.

Let E be a set of examples and $R_B(A)$ a rule. We can define the weight of rule $R_B(A)$ as:

$$\omega_E(R_B(A)) = \frac{n_E^+(R_B(A)) + 1}{n_E^+(R_B(A)) + n_E^-(R_B(A)) + 1}. \tag{3.20}$$

Therefore, the two criteria taken into account to break the tie between two rules are (in the order of appearance):

- The rule with higher weight.
- The rule that was learned first.

3.1.4 Main advantages of SLAVE

The first advantage of SLAVE is that it works with DNF rules, which are widespread in the field of classical machine learning (since they have a greater ability to represent information than the typical rules used in the soft computing field), and keeps the interpretability of the linguistic rules.

In the area of fuzzy sets, SLAVE also incorporates feature selection, an important concept that was already used in classical learning. In particular, SLAVE defines an embedded feature selection, a very useful characteristic for tackling problems with high dimensionality in the number of attributes.

Finally, SLAVE was defined assuming that the rule learning can be performed independently for each class. This independence allows us to implement SLAVE in a parallel way for each class of the consequence variable. Obviously, this results in an improved response time of the algorithm when working with multi-processor computers.

3.2 SLAVE2

3.2.1 Motivation

3.2.1.1 Disadvantages of SLAVE

SLAVE presented two main disadvantages which justified the appearance of SLAVE2:

- While it addresses the issue of the partial relevance of input attributes, the mechanism to detect the irrelevance of an attribute is relatively weak for two reasons. The first one is that the fitness function does not encourage the detection of irrelevant variables, and the second one is that the mechanism to eliminate irrelevant variables is slow and requires increasing the cycles of the genetic algorithm to converge towards a solution.

- The dependence on the λ parameter. This parameter is essential for the algorithm, since it is related to both the number of rules needed to describe a concept and the classification ability of the knowledge acquired. Additionally, this parameter is not easy to estimate a priori, since it depends on the distribution of the examples in the training set.

3.2.1.2 Contributions of SLAVE2

SLAVE2[48, 47, 18] was developed with the objective of improving the behavior of SLAVE when working with high dimensional data and also with the idea of improving the interpretability of the rules obtained during the learning process. To achieve this, it introduces three fundamental changes with respect to SLAVE:

1. A new criterion to penalize examples.
2. A new codification of the individuals in the population.
3. A new evaluation function.

These changes are aimed to correct the problems identified in SLAVE. The first one reduces the dependence on the λ-covering parameter. The second change will allow us to establish a more efficient mechanism to eliminate irrelevant variables and thus to be able to work with databases with a large number of variables. Finally, the third change is motivated by the idea of giving priority to the most simple and interpretable rules.

3.2.2 Description of the main elements

3.2.2.1 A new criterion to penalize examples

SLAVE was not able to appropriately consider the interaction between the rule that was just being learned and the rules already learned. The cause of this problem is that the degree of interaction among rules of different classes is determined by the value of the parameter λ previously described. Let us remember that the value of parameter λ is fixed throughout the process.

The main idea implemented in SLAVE2 to fix this problem was to provide some flexibility in the use of the λ-covering parameter. Thus, to consider that an example is covered by a rule not only requires having an adaptation value for the rule higher than parameter λ (SLAVE), it also requires that the adaptation value of a rule with the correct class exceeds the adaptation value for rules of other classes (SLAVE2).

So, the key idea is that now an example has a certain degree of positiveness (negativeness) only if that degree is enough to correctly (incorrectly) classify the example. In this sense, it is necessary to modify the criterion

for penalizing examples: On each stage of the sequential covering strategy, once the first class is learned, the penalization criterion is adapted for each particular example.

The formal description of this penalization model[48] uses the following values. Let $\lambda^{+}_{Rules}(e)$ be the best adaptation between example e and the learned rules that have the same class as the example, and $\lambda^{-}_{Rules}(e)$ the best adaptation between example e and the learned rules that have a different class:

$$\lambda^{+}_{Rules}(e) = max\{U(e,A) * U(e,B) | \forall R_B(A) \in Rules \text{ and } Class(e) = B\} \tag{3.21}$$

$$\lambda^{-}_{Rules}(e) = max\{U(e,A) * U(e,B) | \forall R_B(A) \in Rules \text{ and } Class(e) \neq B\}. \tag{3.22}$$

From these expressions, we can write new definitions for the number of positive and negative examples of a rule $R_B(A)$:

$$n^{+}_{E}(R_B(A)) = |\{(e, U(e,A) * U(e,B))\}| \text{ such that } \lambda^{-}_{Rules}(e) > \lambda^{+}_{Rules}(e) \text{ and } U(e,A) * U(e,\overline{B}) > \lambda^{+}_{rules}(e) \tag{3.23}$$

$$n^{-}_{E}(R_B(A)) = |\{(e, U(e,A) * U(e,\overline{B}))\}| \text{ such that } \lambda^{+}_{Rules}(e) > \lambda^{-}_{rules}(e) \text{ and } U(e,A) * U(e,\overline{B}) > \lambda^{+}_{rules}(e) \tag{3.24}$$

where $Rules$ is a set of rules previously learned, and $U(e,A)$, $U(e,B)$ are the adaptation values between example e and the antecedent A or the consequent B respectively, which were previously defined.

So, an example e of the training set E is positive for a new rule $R_B(A)$ if e was incorrectly classified before selecting $R_B(A)$ and it can be correctly classified using this rule. In the same sense, an example is negative for a rule if the example was correctly classified and the inclusion of the rule misclassifies it.

Now, during the learning process of a certain class B, the examples removed from the training set E are those $e \in E$ that satisfy

$$Class(e) = B \text{ and } \lambda^{+}_{Rules}(e) \geq \lambda \text{ and } \lambda^{+}_{Rules}(e) > \lambda^{-}_{Rules}(e) \tag{3.25}$$

That is, those examples that are correctly classified for class B and their best match for the rules of their class is equal to, or greater than, λ.

3.2.2.2 A new rule codification for the genetic algorithm

The genetic algorithm used in SLAVE can obviously eliminate irrelevant variables. However, the genetic representation of the information used by SLAVE makes more difficult to eliminate a variable than to add it, and the system might take a long time to find a good solution. Consequently, if we want to improve the detection of irrelevant variables, we first need to change the genetic representation. So, our goal consists in obtaining a better representation of the genetic solutions to make a feature selection for each possible rule. The idea is to include information associated to each antecedent variable in order to discover whether the variable should be considered as part of the antecedent of the rule or not.

In this way, the second change of SLAVE was a new codification of rules[47]. This new codification implemented in SLAVE2 tries to improve the capacity for detecting irrelevant attributes. The idea is to expand the representation with a new level to codify the variables in the rule. Now each individual consists of two substructures or levels, one codifying the values and the other one codifying the variables.

Therefore, a genetic algorithm with two levels maintains two different representations: One for determining the subset of relevant variables associated to a particular class (the variable level) and another one (the value level) to find the best variable-value assignation for that class. On each representation, a process of genetic co-evolution is applied, where each level has its own genetic operators, but the goodness of the solution is calculated by considering the collaboration between both levels of the individual.

Thus, the variable level codifies the variables from the initial set that are considered to be relevant for inclusion in the rule. This information is modified during the evolutionary process. The value level codifies the particular values not only for the variables considered relevant, but also for those considered irrelevant, being the latter not considered when calculating the goodness of the rules.

The separation of these learning tasks (variable learning and value learning) has proven to be very efficient for problems involving a large number of variables. When the genetic algorithm obtains the best description for a concept, it can select a subset of variables and start the search process using this information. Moreover, during the evolutionary process, the subsets of selected variables can be changed with the inclusion of new variables or the removal of some of the variables that were initially selected.

The genetic algorithm has a single population, a single selection criterion and a single fitness function, but it works in a different way for each component of the chromosome through different genetic operators.

The next section briefly describes the main components of the genetic algorithm.

3.2.2.3 The genetic algorithm used in SLAVE2

3.2.2.3.1 Representation of a population element

The representation of an individual in the genetic population is encoded using two structures (see Figure 3.6): One codifies the relevance of the variables and the other codifies the variable-value assignments. With this decomposition, the GA representation has a complex chromosome composed of two structures: A variable chromosome and a value chromosome. This division allows us to clearly distinguish between the two different tasks that are simultaneously carried out by the genetic algorithm (search for the appropriate variables and search for the appropriate value assignment), and we can associate and apply the most appropriate set of genetic operators on each structure.

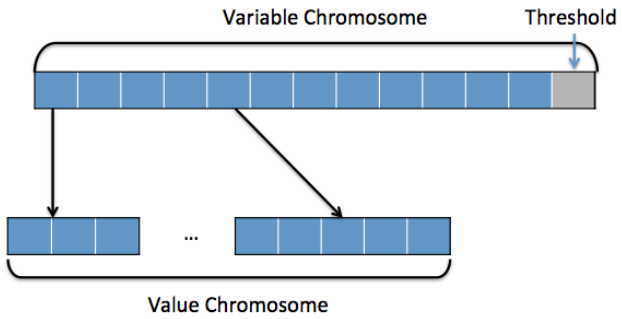


Figure 3.6: Representation of a population individual in SLAVE2.

Let us suppose there are n possible antecedent variables $X_1, \ldots, X_n$, each with an associated fuzzy domain D_i containing m_i components. In order to find the best rule, SLAVE2 fixes a class and then searches for the best antecedent for this class. Therefore, the genetic code must contain information about the relevant variables of the rule and also information about the values of these variables:

- A variable chromosome.
- A value chromosome.

The variable chromosome codifies the relevant/irrelevant variables for the particular rule. It uses a real code with $n+1$ components, in which the i-th element of the j-th chromosome $\tau_C(X_i^j)$ $(i = 1, \ldots, n)$ contains a real value between 0 and 1, representing the relevance degree of the i-th variable with respect to class C, that is, a number indicating the possibility of being a member of the relevant variable set for a rule. The $n+1$ value, named

T_j, is a real value between 0 and 1 representing an activation threshold associated to the j-th chromosome. A variable X_i will be considered as a component of the rule antecedent for a particular class if $\tau_C(X_i^j) \geq T_j$. Otherwise, the variable will be considered irrelevant for the rule. The next subsection explains how to obtain these values for the first population. The genetic algorithm will change these initial values in order to obtain a better estimation of them.

The value chromosome codifies any elements of $P(D_1) \times \ldots \times P(D_n)$ and it is exactly the same one used in SLAVE.

Example 3 *Let us suppose that we have three variables, X_1, X_2, and X_3; the fuzzy domain associated with each one is shown in Figure 3.2. In this case, $m_1 = 3$, $m_2 = 5$, and $m_3 = 2$. Let us consider that the relevance degrees for class C of a population individual are:*

$$\tau_C(X_1) = 0.5, \quad \tau_C(X_2) = 0.7, \quad \tau_C(X_3) = 0.1$$

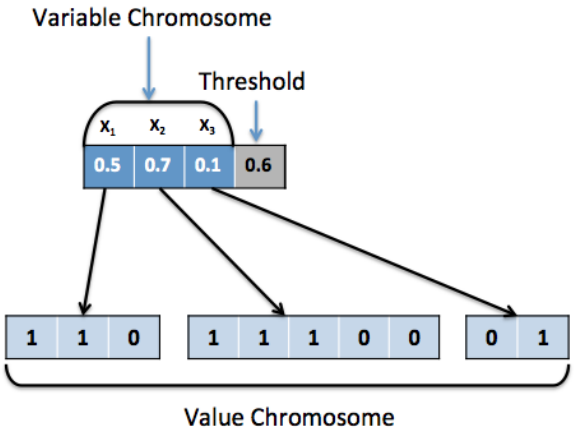


Figure 3.7: Combination variable-value for an individual.

If the combination variable-value for both chromosomes is defined by Figure 3.7 and we take the value 0.6 as the threshold, the antecedent representation would be:

$$X_2 \text{ is } \{A_{21}, A_{22}, A_{23}\}$$

since

- *(110) represents $\{A_{11}, A_{12}\}$, but X_1 is not included in the antecedent since it is not activated in the variable chromosome* $(0.5 \leq 0.6)$.
- *(11100) represents $\{A_{21}, A_{22}, A_{23}\}$ and is included in the antecedent since X_2 is activated in the variable chromosome* $(0.7 \geq 0.6)$.

- *(01) represents* $\{A_{32}\}$*, but* X_3 *is not included in the antecedent since it is not activated in the variable chromosome* $(0.1 \leq 0.6)$.

Obviously, changing the threshold also changes the current description of the antecedent.

3.2.2.3.2 Generation of the initial population

The initial population is generated following a procedure similar to that used in SLAVE for the chromosome value. Thus, each value chromosome is obtained by randomly selecting examples from the class that must be learned and assigning the most specific antecedent that better covers it. This antecedent is made up of only one label for each antecedent variable and the selected label is the one that gives the highest degree of membership for each component in the example. If we consider the domains and variables given in Figure 3.2, the generated chromosome would be

$$(001)(00100)(10)$$

and would correspond to the following antecedent:

$$X_1 \text{ is } A_{13} \quad \text{and} \quad X_2 \text{ is } A_{23} \quad \text{and} \quad X_3 \text{ is } A_{31}.$$

On the other hand, the variable chromosome is built up by using an information function τ_C for each variable with respect to the fixed class on the training examples. The value $\tau_C(X)$ is calculated using the following expression:

$$\tau_C(X) = \frac{I(X, Y = C)}{H(X, Y = C)} \tag{3.26}$$

where the information measure I for variables X, Y is given by the following expression:

$$I(X, Y) = \sum_x \sum_y -p(x, y)\ log_2 \left(\frac{p(x, y)}{p(x)p(y)} \right) \tag{3.27}$$

where x and y are specific values of variables X and Y, C is a fixed class of the consequent variable, and

$$H(X, Y) = \sum_x \sum_y -p(x, y)\ log_2 p(x, y) \tag{3.28}$$

is the Shannon entropy over two variables, where **p** is a probability measure.

The value $\tau_C(X)$ measures the degree of dependence or independence between variable X and the value C of the consequent variable. It can be interpreted as the relevance value of each variable X with respect to class C [47].

When using linguistic variables, it is necessary to define the calculation of this value through the definition of the probability of X when taking a value a_i on its domain $\{a_1, a_2, \ldots, a_s\}$:

$$p(X = a_i) = \frac{1}{m} \sum_{j=1}^{m} \left(\frac{\mu_{a_i}(e_j)}{\sum_{t=1}^{s} \mu_{a_t}(e_j)} \right) \tag{3.29}$$

where m is the number of examples from the training set E, e_j is an example from E, and μ_w is the membership function of the fuzzy set w.

In this formula it is assumed that all the examples are crisp. The bi-dimensional probability is similar to the previous formula, but requires combining the information on two variables using a *t-norm* (denoted by the symbol *).

$$p(X = a_i, Y = b_j) = \frac{1}{m} \sum_{j=1}^{m} \left(\frac{\mu_{a_i}(e_k) * \mu_{b_j}(e_k)}{\sum_{t,h} \mu_{a_t}(e_k) * \mu_{b_h}(e_k)} \right) \tag{3.30}$$

where the domain of variable Y is $\{b_1, b_2, \ldots, b_r\}$ and the t-norm is defined by $a * b = min\{a, b\}$.

Moreover, we need to define an initial value for the activation threshold. T_j takes a random value in the following interval:

$$[min_i \;\; \tau_C(X_i^j), \;\; max_i \;\; \tau_C(X_i^j)].$$

Both $\tau_C(X_i^j)$ and T_j are affected by the genetic operators during the evolution of the genetic algorithm. Therefore, the initial relevance degree, calculated using the above mentioned formulas, is modified during the evolution process until it reaches an appropriate value.

3.2.2.3.3 A new evaluation function based on the simplicity criteria

As mentioned above, the fitness function in SLAVE combines the completeness and consistency measures using a product operator. This fitness function has proven to be very useful for learning on different kinds of problems[43, 50, 44]. However, with this heuristic criterion, many different rules can have the same evaluation function value. The SLAVE genetic algorithm randomly selects one of these rules. SLAVE2 includes a new multicriteria method to discriminate among these rules, instead of using random selection. Among the best rules, the algorithm prefers those which are simpler and more understandable.

This new criterion was included to achieve two different goals:

- To improve the comprehension of the acquired knowledge.

- To obtain a set of rules with a higher degree of accuracy over unseen examples.

The following definitions, proposed in [18], will be useful to introduce new concepts in order to formalize these ideas.

Definition 3.2.1 *Let $R_B(A)$ be a rule with antecedent $A = (A_1, \ldots, A_n)$ and $A_i \in P(D_i)$. A variable X_i, with $i = 1, \ldots, n$, is considered to be irrelevant in this rule if $A_i = D_i$. The number of irrelevant variables of a rule will be denoted as $i(R_B(A))$.*

This definition is based on the treatment of rules used in SLAVE, where the disjunction of adjacent values is taken as the convex hull of the fuzzy labels [43, 45].

This definition allows us to propose the concept of rule simplicity: A rule is simpler than other if it has a lower number of relevant variables. Therefore, the following definition is proposed.

Definition 3.2.2 *Let $R_B(A)$ be a rule. The simplicity degree in variables of this rule is:*

$$svar(R_B(A)) = \frac{i(R_B(A))}{n} \tag{3.31}$$

where n is the number of possible antecedent variables.

The second concept is presented through the following example.

Example 4 *Let X_2 be a variable with an associated ordered domain D_2 (see Figure 3.2). Let $A = \{A_{23}, A_{24}, A_{25}\}$ and $A' = \{A_{23}, A_{25}\}$ be two possible values for X_2. The first value is equivalent to "X_2 is greater than or equal to A_{23}", using the adaptation concept of SLAVE[45], whereas the second one does not have a similar interpretation. If both values are equally appropriate for describing the value of a variable, it is preferred the first one since it is easier to understand. The tie situation is generated by the lack of examples covered by label A_{24}. The preference of the second antecedent is directly related to the generalization principle applied when there is a lack of information.*

Let us consider the definition of stable value:

Definition 3.2.3 *Given a specific value $A_i \in P(D_i)$ for variable X_i, we say that A_i is stable if and only if A_i is composed of a unique sequence of adjacent values of D_i.*

Definition 3.2.4 *Let $R_B(A)$ be a rule with $A = (A_1, \ldots, A_n)$ and $A_i \in P(D_i)$. We define $e(R_B(A))$ as the number of X_i variables required to make D_i an ordered domain and to make A_i or $\overline{A}_i$ a stable value.*

The use of the complementary in the previous definition is justified since a unique sequence of adjacent values in the complementary corresponds to a simple description as NOT A, with A being a stable value. By using this concept, we can define the concept of simplicity regarding the values of a rule.

Definition 3.2.5 *Let $R_B(A)$ be a rule. We define the simplicity degree in values of a rule as:*

$$sval(R_B(A)) = \frac{1 + e(R_B(A))}{1 + p} \tag{3.32}$$

where $p \leq n$ is the number of variables with an associated ordered domain.

With all these elements, we can define a multi-criteria fitness function (consisting of three components) for rule R[18]:

$$\begin{aligned} &fitness(R, E) = \\ &= (\Lambda(R, E) \times \Gamma_{01}(R, E), svar(R), sval(R)). \end{aligned} \tag{3.33}$$

Finally, the selection of the best rule responds to a multi-criteria evaluation function guided by a lexicographical order; that is, the initial criterion (consistency and completeness) is maintained. In case of a tie situation, the simplicity criterion in variables is used; if the tie situation remains, then we appeal to the simplicity criterion in values. Thus, the lexicographical order uses ***(max, max, max)*** as the optimization criteria. That is, to maximize on the first component, then on the second component in case of tie, and finally to maximize on the last component in case of a new tie situation.

Example 5 *Let us suppose we have three variables X_1, X_2, and X_3 with domain D_1, D_2, and D_3 respectively (see Figure 3.2). Let us consider a fixed consequent B and three possible antecedents:*

$$A = (\{A_{11}, A_{13}\}, \{A_{23}, A_{25}\}, \{A_{31}\})$$
$$A' = (\{A_{11}, A_{12}, A_{13}\}, \{A_{23}, A_{24}, A_{25}\}, \{A_{31}\})$$
$$A'' = (\{A_{11}, A_{12}, A_{13}\}, \{A_{23}, A_{25}\}, \{A_{31}\})$$

*with the same value of **(consistency $\times$ completeness)**.*

In this case, the fitness function uses the previous concepts to decide the best antecedent:

- *Antecedent A corresponds to*

$$X_1 \text{ is } \{A_{11}, A_{13}\} \quad and \quad X_2 \text{ is } \{A_{23}, A_{25}\} \quad and$$
$$and \quad X_3 \text{ is } \{A_{31}\}$$

with values $svar(R_B(A)) = 0$ *and* $sval(R_B(A) = \frac{3}{4})$*, since* $\{A_{11}, A_{13}\}$ *is equivalent to NOT* (A_{12})*.*

- *Antecedent* A' *corresponds to*

$$X_1 \text{ is } \{A_{11}, A_{12}, A_{13}\} \quad and \quad X_2 \text{ is } \{A_{23}, A_{24}, A_{25}\} \quad and$$
$$and \quad X_3 \text{ is } \{A_{31}\}$$

with values $svar(R_B(A)) = \frac{1}{3}$ *and* $sval(R_B(A)) = 1$.

- *Antecedent* A'' *corresponds to*

$$X_1 \text{ is } \{A_{11}, A_{12}, A_{13}\} \quad and \quad X_2 \text{ is } \{A_{23}, A_{25}\} \quad and$$
$$and \quad X_3 \text{ is } \{A_{31}\}$$

with values $svar(R_B(A)) = \frac{1}{3}$ *and* $sval(R_B(A)) = \frac{1}{2}$.

Then, the best choice would be antecedent A'*, as it complies with:*

$$fitness(A) \leq fitness(A'') \leq fitness(A').$$

3.2.2.3.4 Genetic operators

As a consequence of the two levels used to codify each individual in the population, one with real codification (variable chromosome) and the other one with binary codification (value chromosome), it was necessary to use different genetic operators for each structure. Taking into account the ones used in SLAVE, the value level inherited them as it uses the same codification. As for the variable level, after some experimental tests the genetic operators that performed better were the uniform mutation, the crossover, and the selection operators. In summary, the genetic operators employed for both levels were:

- Variable level:
  1. Real uniform mutation.
  2. Crossover operator over two points.

- Value level:
  1. Binary uniform mutation.
  2. Crossover operator over two points.
  3. Generalization operator.

It is important to note that we also made some tests using the BLX_α crossover operator at the variable level, but it showed no improvement over the results obtained using the two-point crossover operator.

3.2.2.4 Other components of SLAVE2

3.2.2.4.1 Inference model

Apart from the previously described changes, SLAVE2 includes a modification in the inference mechanism related to the rule weight. In SLAVE, the weight of the rule is used as a secondary criterion for solving conflicts in order to select the winner rule (see Section 3.1.3.4.3). However, SLAVE2 uses the classical inference mechanism of the winner rule, but in contrast with SLAVE, the weight of each rule plays a relevant role in this process. A simple description of this new inference process is shown. Let us consider:

$$Rules = \{R_{B_1}(A_1), \ldots, R_{B_q}(A_q)\}$$

the set of rules, $\Omega = \{\omega_1, \ldots, \omega_q\}$ the weight associated to each rule, and e an example. The examples are assigned the class B_j of the rule $R_{B_j}(A_j)$ verifying

$$j = \text{argmax}_{0 \leq i \leq q}\{U(e, A_i) * \omega_i\}. \tag{3.34}$$

In a similar way to SLAVE, the conflict resolution mechanism used keeps the two criteria mentioned above to break possible ties among the rules, that is,

- The rule with higher weight.
- The rule that was first learned.

3.2.3 Main advantages of SLAVE2

SLAVE2 provides two important advantages with respect to SLAVE. The first one is related to a lower dependency on the λ parameter. This fact causes an increment in the collaboration/competition among rules during the learning process. So, the rules selected previously in the iterative strategy are used to guide the search mechanism and allow reducing inappropriate interactions on the knowledge obtained. The second advantage is a significant improvement of the embedded feature selection, allowing the algorithm to obtain simpler and more general rule sets.

3.3 NSLV

3.3.1 Motivation

3.3.1.1 Disadvantages of SLAVE2

The reduction of the dependency on the λ parameter presents a computational inconvenient, since the algorithm does not admit a parallel version.

Another important problem is that the algorithm maintains a strong bias caused by the need to learn the rules in a particular order. This order is associated with the arbitrary choice of a class before another.

3.3.1.2 Contributions of NSLV

NSLV[46] could be seen as an evolution of SLAVE2 that learns fuzzy rules without fixing the class of the consequent variable.

This simple description is valid, but hides important nuances that must be highlighted. On the one hand, many SLAVE2 drawbacks are solved by not fixing the class during the learning process: The bias due to class selection order disappears; furthermore, it removes the dependence on parameter λ, since the consideration that an example is covered by a rule in its class is determined by the interaction among rules already learned. This parameter is adjusted automatically as new rules are included in the knowledge base.

Moreover, the inclusion of the rule consequent in the search process forces a major redefinition of the algorithm, although the required changes are natural extensions that keep the usual format of SLAVE and SLAVE2.

However, a major problem arises, related to the diversity of the population associated with the genetic algorithm. It is well known that the convergence of GAs can result in a final population where most of the individuals are very similar. To solve this problem, the iterative approach followed in SLAVE and SLAVE 2 resets the initial population to enhance the rule in the next iteration.

However, alternative solutions are possible, and NSLV uses an alternative approach. This approach is based on the use of subpopulations. Thus, NSLV maintains a subpopulation for rules of each class and a combination operator adapted to maintain the diversity in classes. After completing one iteration step, the final population obtained by the GA contains the best rule, but also other promising rules that could be useful and therefore can be used as the starting point for the next iteration.

The following subsections describe in more detail the implementation of these ideas in the algorithm.

3.3.2 Description of the main elements

3.3.2.1 The genetic algorithm used in NSLV

The genetic algorithm of NSLV maintains the basic structure of SLAVE2, with some differences in the representation of individuals, the search mechanism (now we are looking for complete rules), the genetic operators, and the termination condition.

3.3.2.1.1 Representation of a population element

Unlike SLAVE and SLAVE2, NSLV does not learn classes in a specific order. That is why the representation of an individual must include a new level which allows considering the class that is being learned in each moment. This new level is called *consequent level*. So, the complete structure of an individual (and therefore of a rule), would be (See Figure 3.8):

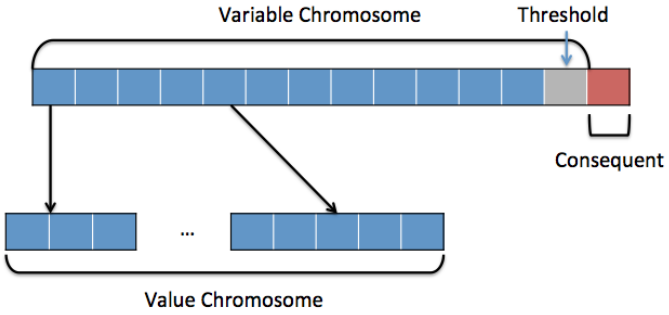


Figure 3.8: Representation of a population individual in NSLV.

The *consequent level* codifies the value of the classification variable of the rule. This level is composed by one gene that is represented through an integer value and is randomly generated in the initial population.

A real example extracted from database *Glass*[3] that could be helpful to better understand the codification of each level in the rule representation is shown in Figure 3.9. The *Glass* database, created by the USA Forensic Science Service, classifies 6 types of glass which can be found in a crime scene, defined in terms of their oxide content (i.e. Na, Fe, K, etc). The first attribute measures the refractive index, while the remaining attributes in the antecedent measure the weight percentage in corresponding oxide.

According to this figure, and focusing our attention on the variable level, we can see that the only variables that exceed the threshold are "*Sodium*" (Na) and "*Magnesium*" (Mg), so they are considered in the rule with their respective value levels. For variable "*Na*", the value level has only one active position related to label "*Very Low*". Regarding to variable "*Mg*", the active label is also "*Very Low*". Thus, this codification corresponds to the rule:

IF Na is {VeryLow} and Mg is {VeryLow} THEN

TypeOfGlass is BuildingWindowsNonFloatProcessed

with **weight 0.93**.

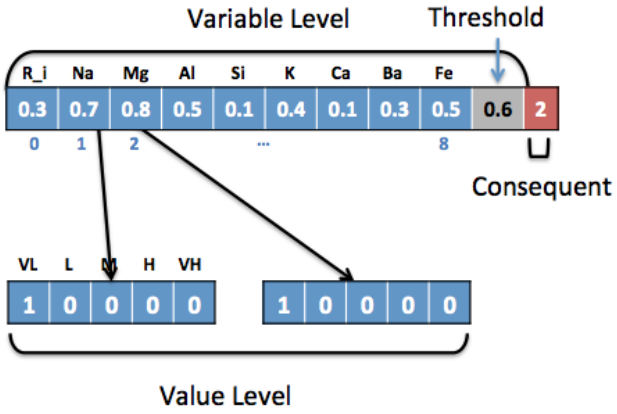


Figure 3.9: Example of a rule codification.

The weight is also encoded in the rule, but since it does not change with the evolution model, it is not included in the representation of a population individual in the genetic algorithm.

So, the previous rule would be easily interpreted as:

"**IF** Sodium is Very Low and Magnesium is Very Low

THEN

TypeOfGlass is BuildingWindowsNonFloatProcessed

with **weight 0.93**"

3.3.2.1.2 Keeping diversity in the genetic population

In order to obtain "good individuals", the genetic algorithm must maintain the diversity in the different classes. That is, it must ensure that there are always individuals of all classes. To achieve this, NSLV uses a population composed of subpopulations or niches (one for each class to be learned) and a modified version of a steady state genetic algorithm whose selection process ensures that no niche disappears from the population.

The selection process is as follows: Two individuals in the population are selected; the crossover operator on each level is applied between them obtaining two new individuals. The mutation operator is used for modifying the new individuals. A standard genetic algorithm replaces these two new individuals with the two worst in the population. This procedure could lead to the elimination of all individuals in a niche. Thus, it is necessary to modify the standard criterion. In this way, the genetic algorithm of NSLV takes an

alternative approach which consists in replacing the two new individuals with the two worst of those subpopulations or niches that are not at risk of being removed from the population. It is considered that a subpopulation is not at risk of being eliminated if it maintains at least m individuals in its niche, where $m = N_{Population}/(N_{classes} + 1)$, $N_{Population}$ is the number of individuals in the population, and $N_{classes}$ is the number of classes evolved in the problem.

This steady state algorithm provides some important advantages. Unlike SLAVE and SLAVE2, which after each iteration had to calculate the initial relevant degree of each variable for defining the variable level for the next population, NSLV defines the variable level of the next initial population as the variable level of the last population. This modification has two advantages:

- A runtime reduction, since the time required for obtaining the initial relevant degree is expensive when there is a large number of examples or variables.
- The last population keeps the best individuals found, and this setting is a good starting point for the next search.

3.3.2.1.3 Genetic operators

As for genetic operators, NSLV uses different ones depending on the level involved. As occurred with SLAVE2, we can distinguish between:

- Variable level:
  1. Real uniform mutation.
  2. Crossover operator over two points.

- Value level:
  1. Binary uniform mutation.
  2. Crossover operator over two points.

- Consequent level:
  1. Integer uniform mutation.

3.3.2.2 Other components of NSLV

3.3.2.2.1 Penalization of examples and termination condition

NSLV uses a new module for the penalization of examples. SLAVE and SLAVE2 removed the examples covered by a rule to avoid considering

them in a later step. In contrast, NSLV marks these examples so they are considered by rules of other classes. As the evaluation of a rule is guided by the examples not covered by any previous one, the marked examples do not affect any rule positively, but they influence the evaluation of rules belonging to other classes.

With respect to the termination condition, the iterative process ends under the consideration of the completeness degree over the whole fuzzy rule set. Thus, if a new rule is added and the completeness degree does not improve, then the learning process ends. The final solution obtained by the algorithm is the complete set of extracted rules, excluding the last one.

3.3.2.2.2 Rule Filtering module

According to our experimental tests, NSLV can add irrelevant rules during the learning process. A rule is considered irrelevant if it is not used at least once for classifying correctly an example. This situation can occur when using an iterative rule learning approach frequently, since it can happen that very specific rules are subsumed by more general rules obtained in later steps. In order to simplify the final rule set, NSLV uses a module for removing irrelevant rules from the learned rule set.

3.3.3 Main advantages and further improvements of NSLV

As already mentioned, NSLV has two major advantages over its predecessors: the elimination of bias in the presentation order of the classes during the learning process, and the elimination of the dependence on the λ parameter of minimum coverage rules. These two advantages have two practical consequences for the algorithm. First, we get knowledge bases with fewer rules, and secondly, the learning time is lower. The reason why we obtain fewer rules is that, when the algorithm learns all the rules (after fixing one class), it must ensure that most of the examples of this class are well covered by the rules. In general, this involves adding additional rules in the knowledge base, in anticipation of possible conflicts with the rules of the classes that are to be learned. This does not happen in NSLV, as it does not have to anticipate future conflicts. The improvement in learning time is related to the reuse of the final population of an iteration as the initial population of the next iteration, since this initial population is now closer to the solutions.

3.4 Experimental study

In this section, the performance of the previously described methods SLAVE, SLAVE2, and NSLV[1] is studied. We have to stand out that these algorithms were originally implemented in C++ language and afterwards they have been translated into JAVA language in order to be included in the KEEL platform [3].

The experimental study has been carried out considering the settings described in Chapter A. The statistical tests applied to perform the comparisons are the Friedman and Iman-Davenport ones in order to find out the significant differences among all the mean values. After that, Holm's test is used to compare the best ranking method against the remaining methods. Apart from these settings, the parameters used for each algorithm in this experimentation are detailed in Tables 3.1, 3.2 and 3.3.

Table 3.1: Specific conditions for SLAVE. NGVL means the number of genes in the value level.

	Specific Conditions SLAVE
Size of genetic population	20
λ parameter	0.8
Number of iterations	500
Mutation prob. (Value level)	0.5/NGVL
Crossover prob. (Value level)	0.1

Table 3.2: Specific conditions for SLAVE2. NGVL means the number of genes in the value level and NAV means the number of antecedent variables.

	Specific Conditions SLAVE2
Size of genetic population	20
λ parameter	0.8
Number of iterations	500
Mutation prob. (Value level)	0.5/NGVL
Mutation prob. (Variable level)	1/NAV
Crossover prob. (Value level)	0.1
Crossover prob. (Variable level)	0.2

It is important to note that NSLV uses a GA based on a steady state approach, keeping niches (one for each class) in the genetic population. These niches are the reason why the size of the population have been increased from 20 to 20*n_class (n_class is the number of classes of the problem) individuals

[1] These algorithms have been recently added to the KEEL platform (http://keel.es/).

Table 3.3: Specific conditions for NSLV. NAV means the number of antecedent variables and *n_class* represents the number of classes of the specific problem.

	Specific Conditions NSLV
Size of genetic population	20*n_class
Number of iterations	500
Mutation prob. (Value level)	0.01
Mutation prob. (Variable level)	1/NAV
Mutation prob. (Consequent level)	0.01
Crossover prob.	1

(compared with SLAVE and SLAVE2). The use of a steady state approach requires changing the probabilities associated to the genetic operators.

The analysis is focused in the study of four parameters related to the fuzzy rule learning algorithms: the accuracy on training and testing sets, the average number of rules and the time employed to obtain the model.

Considering the results given by the Friedman and Iman-Davenport tests regarding to the accuracy on training and test sets (tables 3.4 and 3.6), we note that there are not significant differences among the best ranked algorithm and the rest (p-value > 0.05). Anyway, when looking at the test parameter, the Holm's test indicates that NSLV (the best algorithm in ranking) is close to achieve statistical differences if compared with SLAVE. In this sense, this situation would happen with a significance level of 0.077. With more details, we observe that the best algorithm regarding to the training parameter is SLAVE and the worst is NSLV, which does not correspond to the test results in which NSLV obtains the best results. This means that NSLV is the algorithm that least overfits (Figure 3.10). SLAVE2 has an intermediate behavior, being the best one in training when talking about average values and not far from NSLV in test results (also talking about average values, tables B.1 and B.2).

Table 3.4: Average Rankings of the algorithms (Friedman) and computed p-values by Friedman and Iman-Davenport (accuracy on training set).

Algorithm	Ranking
SLAVE	1.8125
SLAVE2	2.0375
NSLV	2.15
Friedman p-value	Iman-Davenport p-value
0.306895	0.31065665038

Table 3.5: Adjusted p-values (accuracy on training set).

i	algorithm	unadjusted p	p_{Holm}
1	NSLV	0.13121	0.262421
2	SLAVE2	0.314305	0.314305

Table 3.6: Average Rankings of the algorithms (Friedman) and computed p-values by Friedman and Iman-Davenport (accuracy on testing set).

Algorithm	Ranking
SLAVE	2.25
SLAVE2	1.9625
NSLV	1.7875
Friedman p-value	Iman-Davenport p-value
0.1129	0.112263862167

Table 3.7: Adjusted p-values (accuracy on testing set).

i	algorithm	unadjusted p	p_{Holm}
1	SLAVE	0.038606	0.077212
2	SLAVE2	0.433848	0.433848

Table 3.8: Average Rankings of the algorithms (Friedman) and computed p-values by Friedman and Iman-Davenport (average number of rules).

Algorithm	Ranking
SLAVE	2.5875
SLAVE2	2.15
NSLV	1.2625
Friedman p-value	Iman-Davenport p-value
0	0.00000000005

Table 3.9: Adjusted p-values (average number of rules).

i	algorithm	unadjusted p	p_{Holm}
1	SLAVE	0	0
2	SLAVE2	0.000072	0.000072

On the other hand, referring to tables 3.9 and 3.11, we can see that NSLV obtains significant differences in rules when compared with SLAVE and SLAVE2 (p-value $\leqslant 0.05$). In Figure 3.11, it is shown a comparative graphic representing the number of rules obtained by the different methods. With regard to the time parameter, NSLV significantly wins SLAVE and SLAVE2 (again, both p-values are under the α value). Figure 3.12 shows

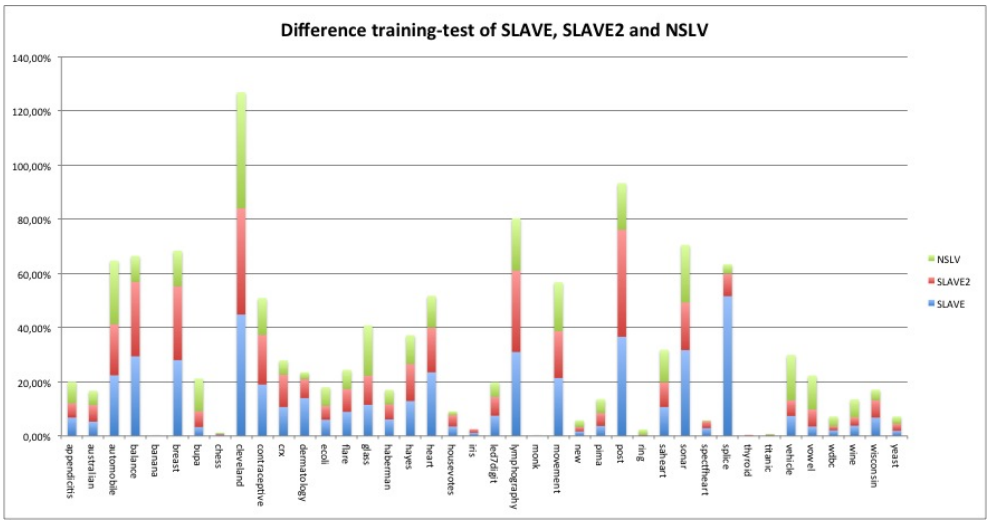


Figure 3.10: Difference training-test for each one of the algorithm involved in the comparison: SLAVE, SLAVE2 and NSLV.

Table 3.10: Average Rankings of the algorithms (Friedman) and computed p-values by Friedman and Iman-Davenport (time employed to obtain the model).

Algorithm	Ranking
SLAVE	2.1125
SLAVE2	2.8125
NSLV	1.075
Friedman p-value	Iman-Davenport p-value
0	0

Table 3.11: Adjusted p-values (time employed to obtain the model).

i	algorithm	unadjusted p	p_{Holm}
1	SLAVE2	0	0
2	SLAVE	0.000003	0.000003

how NSLV achieves better results than the other two algorithms in almost all databases. Its behavior is more regular as it has fewer peaks and its line appears under the lines of SLAVE and SLAVE2 in almost all cases. If we focus on average values (tables B.3 and B.4), we realise that the number of rules achieved by SLAVE is around four times greater than those obtained by NSLV. Moreover, the time employed by SLAVE in order to get the model is almost ten times greater than NSLV, and SLAVE2 invests more than twice the time needed by NSLV to obtain the model. Anyway, these conclusions based on the average results are strongly influenced by the results obtained by the SLAVE algorithm in the *splice* dataset. In this dataset SLAVE

presents very poor results and this fact conditions the average values. If the *splice* dataset would not be considered, the global conclusions would be maintained, but the differences between SLAVE and SLAVE2 would be much closer.

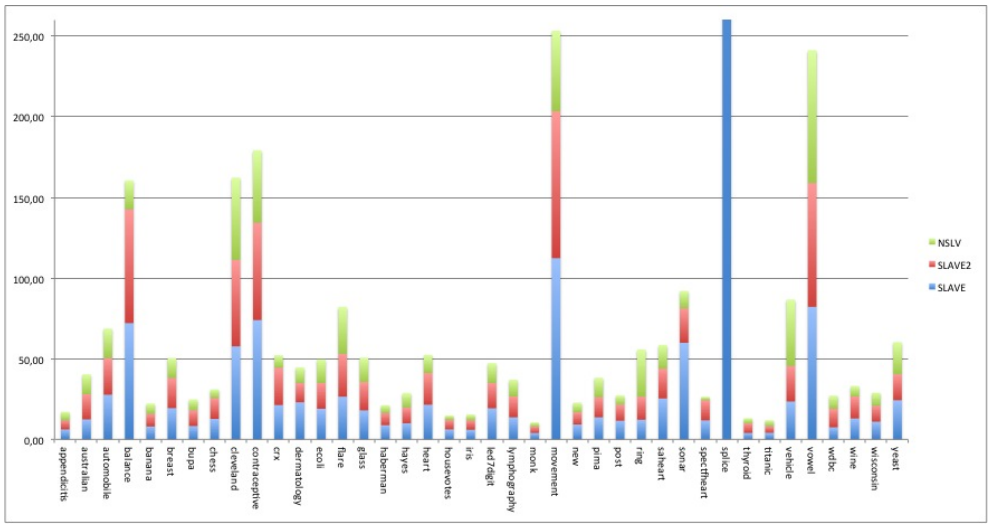


Figure 3.11: Comparison among the number of rules achieved by SLAVE, SLAVE2 and NSLV.

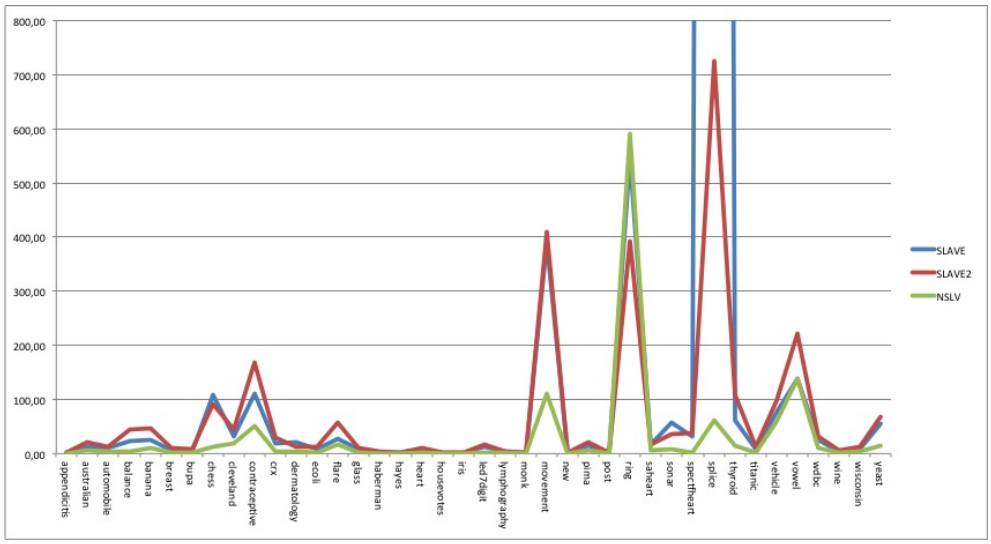


Figure 3.12: Time measured in seconds that SLAVE, SLAVE2 and NSLV employs to obtain the model.

Summarizing, the ability of removing the bias caused by the order in which classes are learned allows NSLV to reduce the overfitting and improves the prediction capability (greater accuracy in testing set), when compared with SLAVE and SLAVE2. Moreover, it also allows reducing the final number of rules of the model in a significant factor. One more important point is that, unlike SLAVE and SLAVE2 (which are generational algorithms), NSLV is a stationary algorithm, which explains why the overall time (referring to mean values), employed to generate the model is much smaller.

3.5 Conclusions

In this chapter we have described the evolution of the SLAVE learning algorithm according to the inflection points during its development.

SLAVE is the oldest of the three proposals, and defines the basic principles and the general framework that were used by later extensions. It was one of the first algorithms that represented knowledge using fuzzy rules, with a structure similar to that of the classical learning algorithms.

SLAVE2 is the next step in the evolution and incorporates two interesting aspects: A new criterion for considering the positivity and negativity of the examples, and model feature selection built in the genetic algorithms.

NSLV is the most recent version, and the first to learn complete rules without having to fix the consequent variable. Furthermore, it uses a steady state genetic algorithm and a niche structure in the population to reduce the learning time.

The experimental study shows that NSLV presents the best overall behavior, being the one obtaining the best results in accuracy and also with less overfitting. It also achieves the best results in number of rules and time with significant differences regarding to the other two algorithms, SLAVE and SLAVE2. Simultaneously, talking about average values, SLAVE2 obtains good results in accuracy, being much better than SLAVE in rules and time parameters.

Chapter 4

Feature construction in a genetic learning algorithm: NSLV-R, NSLV-F and NSLV-FR

4.1 Contribution of feature construction under the learning approach

In Chapter 1, one of the main objectives mentioned in relation to this work talked about the indirect relevance of the input attributes. This fact is considered through an elemental concept called *feature construction*.

In the learning field, we can define the feature construction [11] as the process of constructing new features by the combination, through operators or functions, of the original variables of the problem. The main objective is to extract hidden useful relationships among them in order to better describe the target concept.

Feature construction methods are mainly used to achieve two different goals: on the one hand, for reducing data dimensionality and, on the other hand, for improving the prediction performance. As it is said in [106], the building of automatic feature construction methods keeps being , in nowadays, a challenge. Particularly, we look for methods which:

- Generate new features improving the prediction capability.
- Are efficient from a computational point of view.
- Are easy to generalize.
- Allow easy addition of domain knowledge.

Thus, regarding to feature construction, many different methods have been defined. In order to classify them according to the techniques they use for defining and searching the feature space (as it is proposed in [106]), we can distinguish among three main groups: related to decision trees, related to genetic algorithms and related to ILP (Inductive Logic Programming) based methods.

4.1.1 Decision trees

Most of the early feature construction methods were based on decision trees. In this sense, one of the first works was published by Pagallo [91], in which the learning algorithm FRINGE is proposed. This algorithm uses a *find-feature* procedure for generating new features as boolean combinations of the variables that occur near the fringe of the tree. The set of new features is added to the variable set and the process is iteratively repeated until no new features are added or a maximum number of variables is reached. In [77] it is proposed a system, called CITRE, for performing constructive induction on decision trees using simple domain knowledge. It behaves in a similar way than FRINGE, but differs in the use of the domain knowledge, feature generalization and feature evaluation. Feature construction is performed through the use of disjunctive regions. CITRE collects the feature-value pairs at nodes along these disjunctive regions and proposes all pairs of the binary feature-value pairs as candidate operands. The domain knowledge is limited to simple facts, defining permissible relationships among constructive operands. Regarding to the generalization operator, it works in the following way: if a constructor's operand consists of two features having the same value, a generalized featured appears with its values replaced by a variable. Finally, the evaluation of each feature is measured in terms of the information gained by using the feature to split the whole training set into disjoint subsets.

A later work presented by Hu and Kliber [59] proposes a preprocessing method (independent of the learning algorithm) called GALA, an approach to constructive induction which generates a small number (1 to 3 on average) of new attributes. The method finds new features that are outside the space generated by standard machine learning algorithms. They demonstrate a significant improvement in several artificial and real-world domains and no degradation in accuracy in any domain.

Another related work is shown in [75], where Markovitch and Rosenstein describe an algorithm, called FICUS, which receives as input a set of classified objects, a set of attributes, and a specification for a set of constructor functions that contains their domains, ranges and properties. The algorithm produces as output a set of generated features that can be used by standard concept learners to create improved classifiers. It maintains a set of its best generated features and improves this set iteratively. The algorithm also defines a language for formulating specifications of representation schemes (Feature Space Specifications, FSS), which define the space of the constructed features that will be searched by the generation algorithm.

4.1.2 ILP based methods

According to the definition given in [106], Inductive Logic Programming is used for developing predicate descriptions from examples and background knowledge. ILP based feature construction methods can provide a generalized framework for incorporating background knowledge into the feature generation process. In [105], it is given another definition from a functionally point of view. Thus, ILP can be largely characterized by two classes of programs. The first one, predictive ILP, is concerned with constructing models (set of rules, firs-order variants of classification, regression trees), for discriminating accurately among two set of examples ("positive" and "negative"). The second category, descriptive ILP, is concerned with identifying relationships among the background knowledge and examples, without a view of discrimination. Anyway, many other definitions and more detailed information regarding to ILP can be found in the literature, such as in [82].

One of the early exponents in the use of first-order predicates as features was the LINUS program [71]. Other examples can be found as [105], where Specia et al. investigate the use of ILP for Word Sense Disambiguation (WSD) in two different ways: (a) the construction of models that can be used directly to disambiguate words and (b) the construction of interesting features that can be used by standard feature-based algorithms such as SVMs to build models to disambiguate verbs. They call both kind of models "ILP models" and "ILP-assisted models" respectively.

On the other hand, in [64], Joshi et al. introduce the term **"propositionalization"** as the process of feature construction by an ILP system. They explain that this is usually used as a pre-processing step (in which a large set of possibly useful features are constructed first and then a predictive model is constructed) or by tightly coupling feature construction and model construction (in which a predictive model is constructed with each new feature and only those that result in a significant improvement in performance are retained). In this same work they investigate the applicability of using a randomised search technique for feature construction using ILP. Going on with this idea, in [96], Rückert and kramer introduce a new

margin-based approach to first-order rule learning. The approach uses a novel optimization criterion, Margin Minus Variance (MMV) that is particularly well-suited because it can be calculated in time linear in the number of examples and allows the derivation of error bounds for capacity control. Another work is [70], where Kuželka and Železný describe an algorithm called RelF (Non-redundant Relational Feature Constructor), able to construct a set of tree-like feature set by identifying building blocks (smaller conjunctions) out of which all features can be composed. The main assumption on which the algorithm is built is that the features it constructs, when viewed graphically, correspond to hypertrees while blocks correspond to their subtrees. The feature construction strategy is bottom-up so that the initial set of blocks correspond to all leaves of possible features. Blocks are then progressively combined together with further connecting atoms into larger blocks and eventually into features. Also, in [14], Bresso et al. propose as a first contribution to apply ILP on a logical representation of protein 3D patches corresponding to positive or negative examples of Protein-Binding Sites (PBS), in order to induce a general definition of the PBS concept. As a second contribution, they propose an approach using Formal Concept Analysis for effective interpretation of reached ILP rules with the possibility of adding domain knowledge. Just for finishing with the examples of the use of propositionalization, in [28] it is proposed a fast system for relational learning based on a new form of propositionalization, called Bottom Clause Propositionalization (BCP). Bottom clauses are boundaries on the hypothesis search space and are built from a random positive example, background knowledge and language bias. The idea of using BCP for learning is due to their attempts to represent and learn first-order logic in neural networks.

4.1.3 Genetic algorithms

The use of genetic algorithms for feature construction allows the iterative extraction of new features from the set of initial variables. As a consequence, these new features help to iteratively improve the prediction capability of the model. The main characteristic of evolutionary algorithms is the use of genetic operators (crossover, mutation, etc.), for modifying the population in order to obtain better individuals during the learning process.

In this sense, one of the first works that used feature construction by means of genetic algorithms was [112], in which this strategy is employed by an induction system in order to be tested on difficult texture classification problems.

In [67], Krawiec proposes an extended approach of the framework of GP (Genetic Programming)-based feature construction, where the useful components of features are preserved during an evolutionary run. The function set is composed by +, - , *, %, log, arithmetic comparison operators (LT, GT, EQ), the conditional expression (IF) and the approximate equality op-

erator (EQap). This extended approach proved to statistically outperform the standard approach on some benchmark problems. One more related attempt is shown in [89], where Otero et al. propose a GP algorithm developed for attribute construction. The main motivation is that it performs a global search in the space of candidate solutions. The GP constructs new attributes out of the continuous (real-valued) attributes of the data set being mined. They also evaluate the ability of the attributes constructed by the GP in reducing the error rate associated with the original attributes. They do that by comparing the error rate obtained by C4.5 using only the original attributes with the error rate obtained by C4.5 using both the original and the new attributes constructed by the GP.

Some other works have also been published as [103], where an approach to improve the performance of the induction algorithm C4.5 is presented. Smith and Bull use GP and a Genetic Algorithm (GA) to construct new features. In this context, they use GP individuals consisting of a number of separate trees/automatically defined functions (ADFs) [66] to construct features for C4.5. The GA is then used to select over the combined set of original and constructed features for a final hybrid C4.5 classifier system. In [99], Shafti et al. propose a new fitness function based on Minimum Description Length (MDL). The MDL principle establishes that the optimal solution is obtained by selecting a theory that minimizes the sum of the code lengths corresponding to theory and errors. This fitness is incorporated in MFE2/GA [100] to improve its accuracy. Finally, this system is compared with other ones based on Entropy or error-rate fitness.

Considering different real problems solved through genetic programming, we can find some examples as [5], in which Alfred proposes an algorithm called DARA designed to summarize data stored in nontarget tables by clustering them into groups, where multiple records exist in nontarget tables that correspond to a single record stored in the target table. In [6], a methodology for obtaining accurate and comprehensible classification rules of small and huge datasets is presented. It uses hybrid techniques represented by knowledge discovering. Finally, in [65], Kamath et al. explore the use of Evolutionary Algorithms (EAs) to search in a large and complex feature space. The goal is to obtain features able to significantly improve the classification accuracy of Support Vector Machine (SVM). This approach is evaluated on the difficult problem of DNA splice site prediction.

Once we have seen different ways in the use of feature construction, we have to say that we have focused our efforts in the implementation of this technique through genetic algorithms. So, next sections are devoted to describe two methods that include feature construction by means of relations and functions as a part of the antecedent of a fuzzy rule.

4.2 Learning fuzzy relational rules

4.2.1 Motivation

In the previous section we analysed the main contributions of the feature construction approach in the learning field. Once we are inside of feature construction, we can deal with this technique through different methods. As we are addressing in this section, now we are focused in the use of relations in the antecedent of a fuzzy rule. This kind of rules, also called Fuzzy Relational Rules (FRR), is characterized by making flexible partitioning of the input space. Thus, the relations as a part of a fuzzy rule let us to obtain models with a good trade-off between accuracy, interpretability (at rule level) and simplicity in the description of the problem.

Related to this topic, we can find several works using relations as a method of feature construction. In [38] Gaweda and Zurada present an approach to fuzzy rule-based modeling of nonlinear system from numerical data. They introduce interpretable relational antecedents that incorporate local linear interactions between the input variables into the inference process. A later study is proposed in [1] by Akbarzadeh et al. in which an evolutionary system for derivation of fuzzy classification rules is presented. The system uses two populations for discovering classification rules. The first one for encoding and the second one keeps the membership functions definition for fuzzification of continuous attributes and the relational operators between the attributes that have the same type.

Some more recent works can be cited as [97] where Scherer defines a basic relational fuzzy model to be applied on relational neuro-fuzzy systems. These kind of systems are versatile and can be used in various tasks of classification, modelling and prediction. In [87] Nojima and Ishibuchi examine the use of fuzzy relational conditions with respect to the relation between two input variables. They define three simple fuzzy relational conditions for pattern classification to demonstrate the usefulness of generated fuzzy relational rules in their multiobjective genetic fuzzy rule selection.

According to the ideas previously exposed, in next section we briefly describe the most important components of a learning algorithm that uses relations in the antecedent of fuzzy rules.

4.2.2 The learning algorithm NSLV-R

The main contribution of the relations in the antecedent of fuzzy rules is to extract useful information from the comparison between the input variables. They improve the interpretability from a human point of view and also increase the simplicity of each rule (reducing the number of conditions in the antecedent). In this sense, in order to incorporate this functionality, a learning algorithm called NSLV-R [15, 49] was developed. Thus, NSLV-R

is a fuzzy rule learning algorithm that inherits the basic properties of its predecessor NSLV [37] and introduces relations among the initial variables in the antecedent of a rule.

This algorithm introduces three main changes with respect to NSLV:

- The genetic representation of a rule.
- An index set to handle the most interesting relations, called Catalog of Relations (CR).
- A new criterion for calculating the number of positive and negative examples to a rule.

These changes are devoted to manage the relations in the learning process. The first one is associated to a new substructure to encode the active fuzzy relations appearing in the rule. The second change is related to the storage of the most interesting relations able to be considered in a rule. Finally, the third one is needed in the definitions of consistency and completeness in order to be adapted to the new structure of the rules.

Thereby, an example of the kind of rules we are interested in, is shown below:

IF X_1 is *aproximately_equal_to* X_2 is *Low* and X_3 is $High$ **THEN** Y is *Low*

where *aproximately_equal_to* is a fuzzy relation that would generate the partition shown in Figure 4.1.

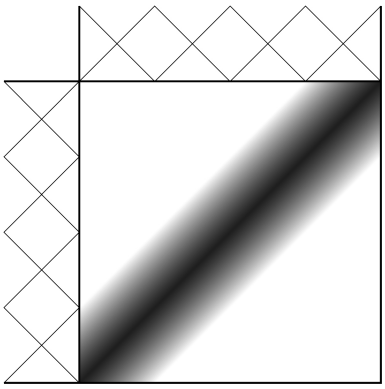


Figure 4.1: Partition generated by the fuzzy relation *X is aproximately equal to Y*. This figure has been taken from [15].

4.2.2.1 First change: The genetic representation of a rule

In Section 3.3, we described, among others, the different levels in the genetic representation of a rule for NSLV. Basically, we had three: the variable level,

the value level and the consequent level. Now, NSLV-R adds a new level to encode the active fuzzy relations called *relation level* (Figure 4.2).

- *The relation level.* This level represents the relation set included in the antecedent of the rule. Each gene codes a possible relation of the catalog of relations (CR). It uses an integer coding where 0 indicates "no relation" and any other positive value refers to the index of the relation in the catalog. In the initial population, all individuals begin with all the genes of this level with zero value, that is, at the beginning of the genetic process, no relations are considered in the chromosome.

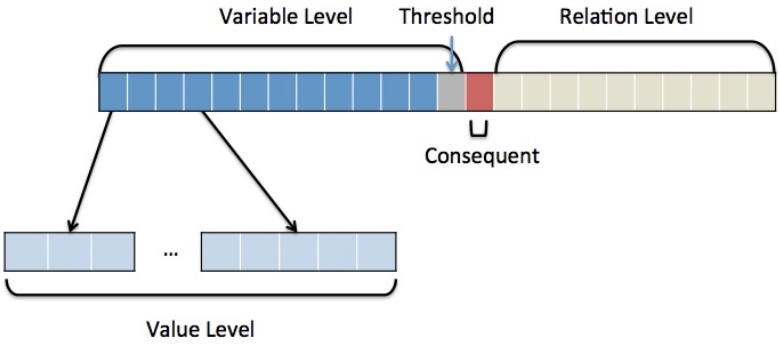


Figure 4.2: The genetic coding in NSLV-R.

4.2.2.2 Second change: Catalog of Relations (CR)

The Catalog of Relations can be seen as a storage structure that works as an index set. The idea associated to the use of this structure is to contain the most promising relations selected after a filtering process. The filtering mechanism is carried out in two stages. Thus, in the first stage relations are discarded through an approach based on the overlapping between the domains. Every two variables with continuous domains (discrete ones are not considered) are compared. If the degree of overlapping exceed a threshold, then any relation between both variables would be candidate to be part of the CR.

In the second stage, the filter depends on an information measure in order to select only the most relevant relations. This information is associated to each candidate relation selected in the previous stage.

In Figure 4.3, it is shown an example of the representation of the *relation level* together with the content of the CR.

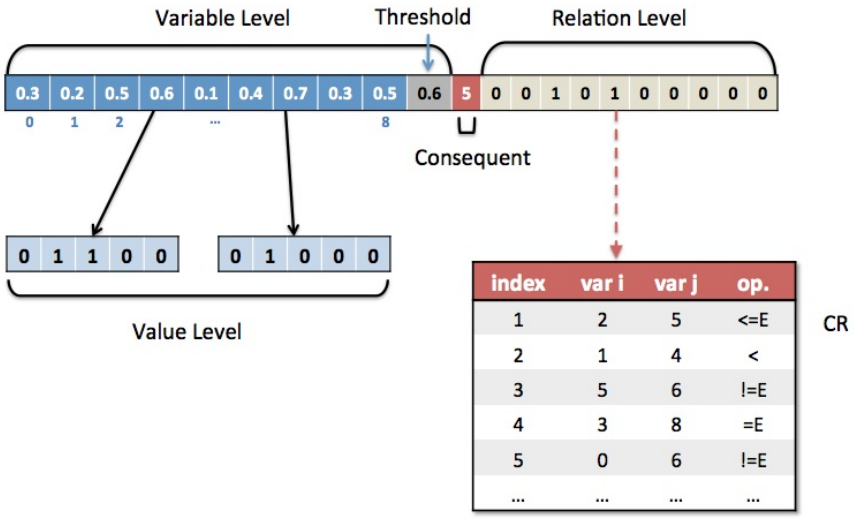


Figure 4.3: The genetic coding in NSLV-R and the CR.

4.2.2.3 Third change: Criterion for calculating the number of positive and negative examples to a rule

Taking into account the change in the representation of an individual of the population, the calculation of the number of positive and negative examples to a rule must be adapted in the definitions of consistency and completeness.

So, given a fuzzy relational rule, $R_B(\tilde{\mathrm{A}}, H)$, the number of positive examples to this rule is defined as the cardinal of the fuzzy set of positive examples to the rule. This set is defined using a membership degree to each example over the concept "to be positive" to a rule. This membership degree is defined by

$$\tau(\tilde{\mathrm{A}}_1, \ldots, \tilde{\mathrm{A}}_n, H) \wedge \frac{B(y)}{max_{B'} B'(y)}$$

representing the simultaneous adaptation of the example to the antecedent and the consequent of the rule. The number of negative examples is the cardinal of the fuzzy set of negative examples to the rule. The membership degree of each example to this set is defined by

$$\tau(\tilde{\mathrm{A}}_1, \ldots, \tilde{\mathrm{A}}_n, H) \wedge \frac{max_{B' \neq B} B'(y)}{max_{B'} B'(y)}$$

representing the adaptation to the antecedent and any of the other possible values of the consequent variable different to that considered in the rule.

In the previous formulation $\tilde{\mathrm{A}}_i$ represents a set of fuzzy values on the universe U_i, B is the consequent value and $H \subseteq CR$ is an index subset

defining the concrete relation part of the rule, where:

$$\tau(\tilde{A}_1, \ldots, \tilde{A}_n, H) = \tilde{A} \wedge \{\wedge_{(i,j,k)\in H} R_k(x_i, x_j)\}$$

and

$$\tilde{A} = \tilde{A}_1(x_1) \wedge \ldots \wedge \tilde{A}_n(x_n).$$

Taking the previous information under consideration, the final rule model used in this approach would be given by:

$$\text{IF } X_1 \text{ is } \tilde{A}_1 \wedge \ldots \wedge X_n \text{ is } \tilde{A}_n \wedge \{\wedge_{(i,j,k)\in H}[(X_i, X_J) \text{ are } R_k]\}$$

$$\text{THEN } Y \text{ is } B$$

being R_k the k-th relation in the catalog involving the antecedent variables X_i and X_j.

These are, basically, the most significant changes that NSLV-R introduces with respect to NSLV. Anyway, in Section 4.4, we will explain with more details the mechanism used in the learning process to handle the relations. In Figure 4.4, we can see a global view of how NSLV-R works.

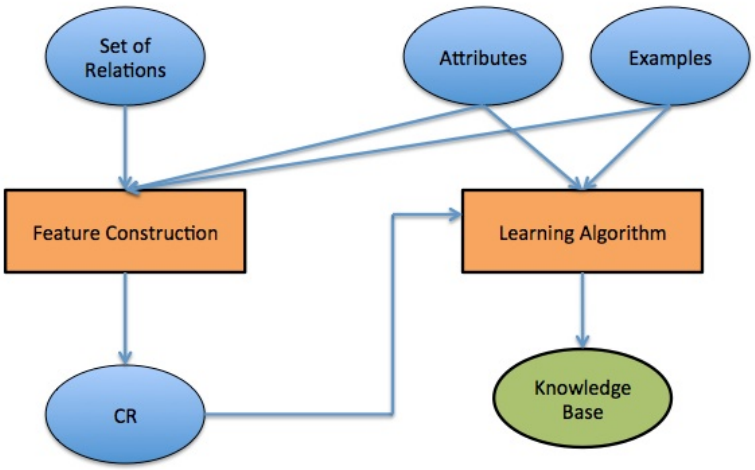


Figure 4.4: Global view of NSLV-R.

4.3 Learning fuzzy rules with functions in the antecedent

4.3.1 Motivation

The success of any classification algorithm depends on its ability to represent any inherent pattern in the data set and, hence, depends on the set of

predictive attributes available, or its attribute vector. If the set of available attributes does not include one or more powerfully predictive attributes, this will limit the performance of classification algorithms that are unable to combine these attributes in any way. Thus, one approach to overcome this problem is to allow the induction process the flexibility to identify and construct these powerfully predictive combinations.

We have just seen that the use of relations in the antecedent of a fuzzy rule allowed to obtain models with a good interpretability and accuracy but maintaining the simplicity in the description of the problem. This is a consequence of the flexible partitioning of the input space that the FRR make.

Now, we are interested in addressing the feature construction through the use of functions in the antecedent of fuzzy rules. We want to explore the influence of adding to the antecedent of a fuzzy rule non-trivial information derived from the combination of the initial set of variables.

In the literature, it is possible to find some examples in which this way of feature construction is used. For instance, in [110] Tackett developed a GP to construct new features from features extracted from segmented images relating to target identification. Classification trees were constructed using the GP features and compared with more conventional methods. The GP achieved a higher performance with reduced computational effort. In [83], Muharram and Smith use four different fitness functions in the attribute construction with Genetic Programming (GP), applying four different classifiers, thoroughly analysing the evolved variables and resulting trees and, finally, comparing the GP used in feature construction with GP used as a "decision tree classifier.

On the other hand, in [25] Dor et al. presents a feature discovery approach called FEADIS, which tries to iteratively extract new features formed by different mathematical functions like *ceil*, *mod*, *sin*, etc. The main objective is to improve the learning performance of the extracted features. They experimentally demonstrate that FEADIS is an efficient resource when working across datasets with both nominal or numeric target feature and periodical datasets.

Commonly, it is easy to find many problems in which the number of functions and also of attributes can be very large. As a consequence, it could be interesting to use a mechanism to reduce possible combinations between initial variables through the set of defined functions. The main idea used in this proposal is to define a filter model, based on the use of information measures so that the genetic algorithm only explores the particular new features that may be more interesting to the final identification of the system. Therefore, a key problem in the scope is the complexity added by the use of a new model of rule. Initially, the number of functions and attributes that could be used in these functions would make that the possible number of particular functions to be taken into account by the algorithms would be

too large, and the benefit of using new attributes would not be profitable given the time required. Thus, previously to use new features in the learning algorithm we need a procedure to extract the most relevant combinations of functions and features for the particular problem. This procedure makes use of an information measurement that evaluate the relevance of each new attribute in relation to the consequent variable. Then the procedure selects the most relevant new variables, which are then supplied to the learning algorithm, which performs a special treatment of these variables. The basic idea is to include a new variable only when there is a clear improvement caused by this inclusion. The set of new attributes is called *Catalog of Functions (CF)*. Since this set is obtained in a previous step to its use by the learning algorithm, we are really using a filter feature construction. Finally, the learning algorithm takes the Catalog of Functions and the training example set for obtaining a knowledge base representing the system. Figure 4.5 shows a graphical representation of the procedure.

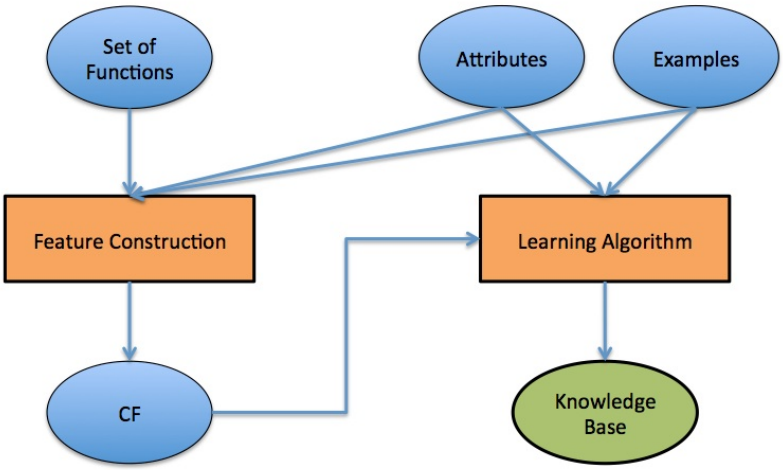


Figure 4.5: Filter feature construction.

4.3.2 Inclusion of functions in the antecedent of a fuzzy rule

According to the ideas previously exposed, now we want to improve the capacity of knowledge representation of fuzzy models allowing that a function can be included as part of the description of a rule. To extend the rule model is easy since a function can be considered like other new variable. For example, let's consider $\{X_1, X_2, X_3\}$ three antecedent variables and Y a consequent variable. The fuzzy domain for all the variables is described in Figure 4.6. If we want to use the following rule (the definition of the final rule model will be given in (4.7)):

IF $SUM\{X_1, X_2\}$ is Low and X_3 is $VeryHigh$ **THEN** Y is $Medium$

it is necessary to define the fuzzy domain of the new variable $SUM(X_1, X_2)$ representing X_1+X_2, and Low must be a value of this domain. Obviously, we can interpret that $Z = SUM(X_1, X_2)$ is a new linguistic variable, therefore, to work with functions in the antecedent of the rule does not modify the traditional inference process of the linguistic rules.

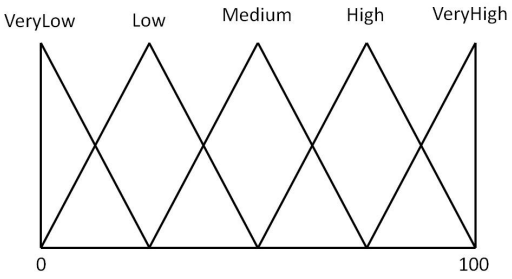


Figure 4.6: Domains associated to the variables of the example

In order to define the fuzzy domain of the new attributes the general process is so simple. Let's suppose that the following variables X_i, X_j, are defined in a fuzzy bounded domain

$$U_i = [inf_i, sup_i]$$

and

$$U_j = [inf_j, sup_j]$$

respectively. The range for the resultant variable Z, that represents the function $f(X_i, X_j)$, will depend on the operation selected in each case, but in general, it could be defined as

$$[min_{x_i \in U_i, x_j \in U_j} f(x_i, x_j), max_{x_i \in U_i, x_j \in U_j} f(x_i, x_j)].$$

where U_i and U_j are the universe for X_i and X_j respectively. For this range, we consider a fixed number of uniformly distributed linguistic labels in order to define the domain of the new variable (in this work, this value is fixed to five). The fuzzy domain of the $SUM(X_1, X_2)$ of the previous example is defined in Figure 4.7.

4.3.3 The feature construction filter

The feature construction filter tries to find a set of new attributes that allow us to acquire more information about the problem. As it is shown in Figure

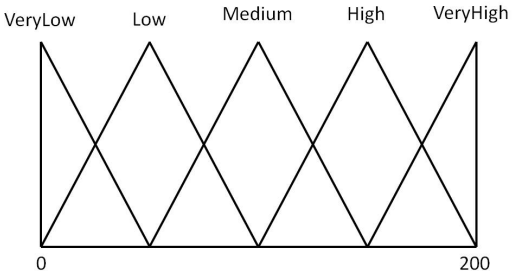


Figure 4.7: Domain of $Z{=}SUM(X_1, X_2)$ variable

4.5, the input of this module is the initial variables set (V), the training example set (E) and a set of predetermined functions (F). When we set a number of general functions that the algorithm can use, we are really reducing the search space. In this process is very important a previous analysis of the database from which we want to learn.

The new variables are constructed applying the functions over some of the initial input variables. Although the general model could work with any kind of functions, from now on we will only work with binary functions, that is, functions over two variables.

In principle for a general function, as the sum of variables, we could consider any subset of two variables obtained from E. However, applying the sum of any two variables may not provide an easy interpretation. So, in many cases it is useful to consider the units associated with each variable, and apply functions such as addition, only over variables that have the same type of units. A simple example is that if we take as the first variable the age of a person and as the second variable the weight of this person, to consider the sum of these two variables probably has no special interest, however, other combinations of variables could be more interesting. Of course the use of units depends of the particular function. No doubt the information provided by an expert could be critical to limit the combination of variables considering their units. In this sense, we consider a value $Unit(X)$ associated with each variable involved in the problem that tries to represent the concept of unit. This value, coded by an integer, permits to the filter process to determine if a subset of variables could be combined by a determined function. So, $Unit(X) = 0$ indicates that X could not be combined with any other variable. For other values of $Unit(X)$, X could be combined with other variables by a function f, if the subset of variables, where X is included, satisfies the restriction considered by f with respect to the units. For example, the addition function $SUM(X1, X2)$ is applicable if $Unit(X1) = Unit(X2)$ being $Unit(X1) \neq 0$ and $Unit(X2) \neq 0$.

When the combination is possible, the relevance of each new attribute is evaluated using an information measure on the examples from the training

set. The following code describes the process:

- FOREACH **f** in **F**
  - FOREACH subset **S** of **X** that could be combined by f
    - $info(f(S), E)$ is a measure of the relevance of the f function with S variables, calculated on the training set E.
    - IF $(info(f(S), E)) > \vartheta$, being ϑ a threshold that establishes the minimum value for a function to be considered THEN $f(S)$, is added to the provisional set of new attributes E.
- ORDER all the new variables obtained in previous step using the $info$ function, and eliminates all but the top N. Where N is an user parameter that established the maximum number of new variables to be used by the learning algorithm.
- RETURN the set of new attributes $\hat{X}$, where $|\hat{X}| \leq N$.

To implement this process we will detail below how we will obtain the measure $info(f(S), E)$ and the value of the parameter ϑ.

Let F be a set of previously defined (by an expert) functions. For each one of these functions we consider a subset of two input variables. In order to know the relevance degree of the function over the set of examples we use an information measure (the same as that used in Chapter 3, Section 3.2.2.3.2). This measure is detailed below [69, 117]:

$$\rho(X, Y) = \frac{I(X, Y)}{H(X, Y)} \tag{4.1}$$

where

$$I(X, Y) = \sum_x \sum_y -p(x, y)\ log_2(\frac{p(x)p(y)}{p(x, y)}) \tag{4.2}$$

and H(X,Y) is the Shannon entropy over two variables, defined as

$$H(X, Y) = \sum_x \sum_y -p(x, y)\ log_2\ p(x, y). \tag{4.3}$$

The ρ measure estimates the dependence between two generic variables X and Y in the following way: values of $\rho(X, Y)$ close to zero determine a high degree of independence of both variables, whereas values close to one demonstrate a high degree of functional dependency between them.

In our case, where we have a new candidate variable, for example,

$$Z_k = f_k(X_i, X_j)$$

obtained by the use of the function $f_k \in F$ and the particular initial variables X_i and X_j both belong to V, then, we calculate:

$$\rho(Z_k, Y) = \frac{I(Z_k, Y)}{H(Z_k, Y)} \tag{4.4}$$

where Y is the classification variable.

In this way, the measure $info(f(S), E)$ is exactly that described in (4.4). Once we have this information measure we compare it with a threshold (ϑ) that establishes the minimum value for a function to be selected. The comparison method is as follows. We consider the value M_I as the maximum information given by any possible initial variable and it is defined by:

$$M_I = max_p(\rho(X_p, Y)), \forall p \in \{1...n\} \tag{4.5}$$

being n the number of initial variables and Y the classification variable. Then, the threshold value is calculated with the following expression:

$$\vartheta = \alpha * M_I$$

being $\alpha \in [0, 1]$, that is, the threshold is taken as a percentage of the maximum relevance value for the initial variables. In the experimental studies we have considered $\alpha = 0.9$. Using this parameter, the comparison expression used in the previous code to consider the variable Z_k is:

$$\rho(Z_k, Y) > \vartheta. \tag{4.6}$$

So, if the function's information measure exceeds the established threshold, this new variable is candidate to be added to the final set of new attributes. Finally, this set of new attributes is returned at the end of the process taking the best N variables.

The output of the previous algorithm is used to define a new structure that we will call "Catalog of Function" (CF). In a formal way it is an index set where each entry is an unique relevant function (the function and the particular variables involved) to be considered by the learning algorithm. For a given i, j, k, we define an entry in the CF as a three-position vector where the first and the second values make reference to the variables involved in the function and the last one is the function applied to these variables. An illustrative example for the previous given values would be $f_k(X_i, X_j)$. In this way, we will be able to associate to this function an integer representing the index of the particular function $f_k(X_i, X_j)$ in the catalog. This integer can be described by means of the expression $\beta(i, j, k)$, because it is ordered in the catalog. We also define

$$Z_{\beta(i,j,k)} = f_k(X_i, X_j),$$

as a new variable resulting of applying the function f_k over the variables (X_i, X_j).

Now we can define the new proposed model of fuzzy rules as follows:

$$\text{IF } X_1 \text{ is } A_1 \wedge \ldots \wedge X_n \text{ is } A_n \wedge \{\wedge_{(i,j,k)\in H} f_k(X_i, X_j) \text{ is } V_k\} \text{ THEN } Y \text{ is } B$$
$$\text{with weight w}$$

where A_i are the antecedent values, B is the consequent value, $H \subseteq CF$ is an index subset defining the specific function participating in the rule and w the weight of the rule. On the other hand, V_k is a fuzzy value of the fuzzy domain of $f_k(X_i, X_j)$.

As we have been managing an extended version of DNF rules, we are able to assign a set of values from its domains to each antecedent variable, so now the new rule definition is:

$$\text{IF } X_1 \text{ is } \tilde{A}_1 \wedge \ldots \wedge X_n \text{ is } \tilde{A}_n \wedge \{\wedge_{(i,j,k)\in H} f_k(X_i, X_j) \text{ is } \tilde{V}_k\} \text{ THEN } Y \text{ is } B$$
$$\text{with weight w} \tag{4.7}$$

with $\tilde{A}_i$ being a set of fuzzy values on universe U and $\tilde{V}_k$ a set of fuzzy values defined on the fuzzy domain of $f_k(X_i, X_j)$.

From now on, we will refer to this kind of rules as Fuzzy Rules with Functions in the Antecedent (**FRFA**). It is important to mention that the new variables $Z_{\beta(i,j,k)}$ will have their own domain depending on the domains of the variables involved in the function, as was described in the previous section.

Actually, we use a model of FRFA which includes a weight on each rule. This weight, a value in [0, 1], is interpreted as a measure of its prediction capability, and its value is calculated as the percentage of examples correctly covered by the rule divided by the total number of examples. This value depends on the particular rule and the training set only, and keeps fixed during the evolutionary process. In the inference process, the weight of each rule modifies the adaptation to the example with the antecedent of the rule by means of a product operation. The winner rule is the one that obtains the highest value in this adaptation process.

4.3.4 The learning algorithm NSLV-F

One of the major changes that NSLV-F introduces is related with the codification of a rule. This new codification together with the use of the CF allows to handle a reduced number of combinations between the original variables. Thus, we propose a different treatment of the new variables in relation to the original ones. With this purpose, we maintain the variable and value levels of the initial variables and we add a new substructure called *function level* that encodes the active functions that appear in the rule.

- *The function level.* This level represents from the total set of functions, those that are part of the antecedent of the rule. The maximum number of functions that can be included in a single rule is defined by a parameter. In the experiments we have considered up to 10 functions in each rule. This function level uses an integer coding where 0 indicates "no function considered" and any other positive value will refer to the index of the function in the CF (Fig. 4.8).

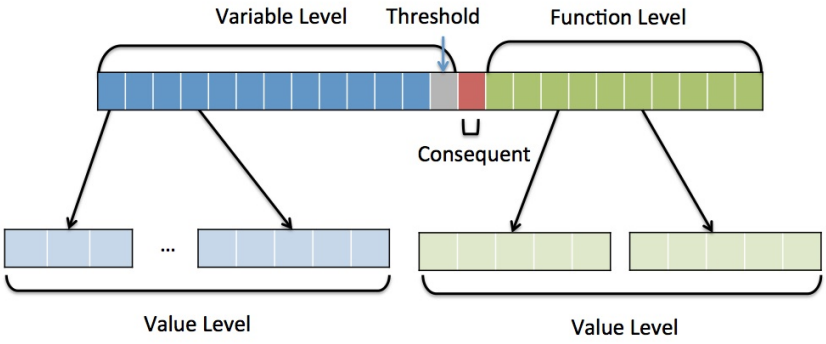


Figure 4.8: A rule representation in NSLV-F

Regarding to genetic operators, we keep the same operators for the variable, value and consequent levels as used in NSLV. Due to the integer codification of the function level, we use two points crossover and mutation operators for integer coding. The mutation operator defined for this level has an associated probability. This probability establishes whether a value of the structure is modified (for experimental studies this value has been set to 0.2). When the previous fact occurs, a position of the level is randomly selected. If that position has zero value then a new random value is generated for selecting a new function of the CF. If the position has a different value from zero, then zero value is assigned to that position.

On the other hand, we also have a new value level associated to the function level which uses a binary coding, so we define the same two genetic operators that were defined for the value level associated to the variable level. That is, two points crossover and binary uniform mutation. Obviously, the level is associated to the particular fuzzy domain of each function.

In the initial population all individuals start having the genes of their function level with zero value. That means that at the beginning of the genetic process no functions are considered in the chromosome. The value level associated to each function gene is randomly assigned.

4.4 Learning fuzzy rules with relations and functions

Considering the assumptions given in the previous sections, the purpose now is to present an integrated model that combines the main ideas exposed above.

Thus, the objective in this case is to learn fuzzy rules with relations and functions in the antecedent so that it is possible to exploit the advantages of both methods in order to extract more information from the original variables.

Regarding to the performance of this model, the learning algorithm takes as inputs the initial variable set, the training example set, the catalog of functions (CF) and the catalog of relations (CR). It also considers a set of pre-defined functions together with another set with the pre-defined relations. Initially, with the set of pre-defined relations and functions, the original variables and the example set, the feature construction module looks for those relations and functions that applied over each two original variables give the best information measure. The selected relations are then stored in the CR while the most promising functions are stored in the CF. Then, the learning algorithm iteratively extracts one rule for a selected class and repeat the process while the new extracted rule improves the last one and till no more rules are needed to cover the examples of that class.

In Figure 4.9 it is summarized the previous process. There are two separated stages: the first one devoted to feature construction, in which both catalogs are built, and the second one, where the learning process takes place.

The catalogs keep the same structure and behavior as commented in subsections 4.2.2.2 and 4.3.3. Taking it into account, it is possible to define a new model of rule mixing both structures in the following way:

$$\text{IF } X_1 \text{ is } \tilde{A}_1 \wedge \ldots \wedge X_n \text{ is } \tilde{A}_n \wedge \{\wedge_{(s,r,t)\in Q}[(X_s, X_r) \text{ are } R_t]\}\wedge$$

$$\wedge\{\wedge_{(i,j,k)\in H} f_k(X_i, X_j) \text{ is } \tilde{V}_k\} \text{ THEN } Y \text{ is } B \text{ with weight w}$$

where $\tilde{A}_i$ are the antecedent values, B is the consequent value, $Q \subseteq CR$ and $H \subseteq CF$ are index subsets defining the specific operations (relations or functions), participating in the rule, R_t is a relation applied over the initial variables X_s and X_r, and $\tilde{V}_k$ is a fuzzy value belonging to the domain of the function $f_k(X_i, X_j)$. As occurred with the previous models, w is a value in [0, 1], that indicates the weight associated to the rule and is interpreted as a measure of its prediction capability. The expression is calculated through the definition of the number of positive (n^+) and negative (n^-) examples given in 3.2.2.1 as:

$$w = (n^+ + 1)/(n^+ + n^- + 1)$$

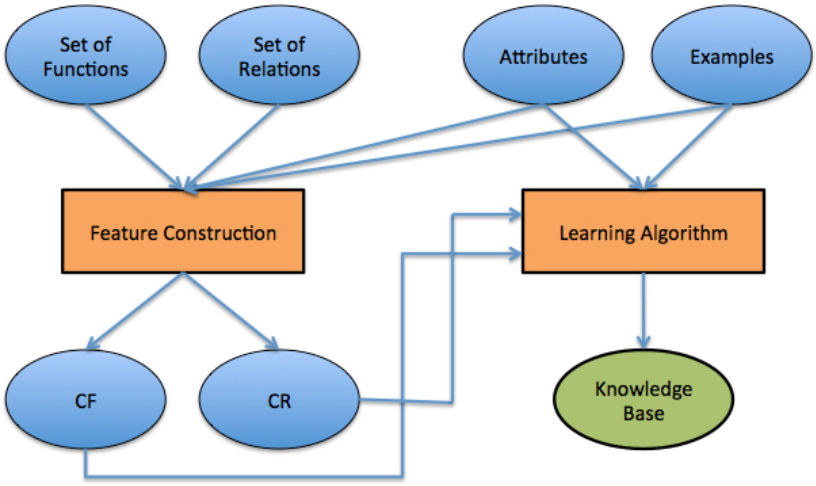


Figure 4.9: General view of the feature construction and learning process.

In order to clarify the structure and the representation of the different levels in a rule, Figure 4.10 shows an example extracted from *Glass* database A.1.

According to this figure, in the variable level the only variables exceeding the threshold are "*Aluminium*" (Al) and "*Calcium*" (Ca). So, these are the only one considered in the rule with their respective value levels. For variable "*Al*", the value level has two active positions, tha is, labels "*Low*" and "*Medium*". The same occurs with variable "*Ca*", which has the active label "*Low*". Thus, the first part of the rule would be:

IF Al is {Low, Medium} and Ca is {Low} ...

In the relation level, there is only one position with value different to zero which indicates the corresponding entry in the CR (in this case, the first entry). Now, the rule would be:

IF Al is {Low, Medium} and Ca is {Low} and Mg $\lessapprox$ K ...

With regards to the function level, the active positions indicate the entries in the CF (the third entry in the example). As functions can be considered new variables, they have their own value level. In this case the value level presents four active labels: "*VeryLow*", "*Medium*", "*High*" and "*VeryHigh*". The rule would change as follows:

IF Al is {Low, Medium} and Ca is {Low} and Mg $\lessapprox$ K and

SUBT (Na, Mg) is {VeryLow, Medium, High, VeryHigh} ...

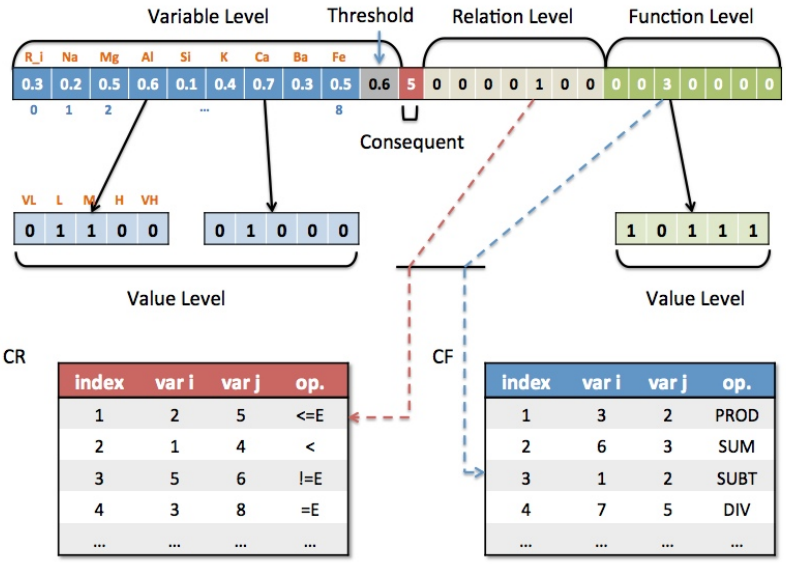


Figure 4.10: Example of a rule coding for *Glass* database.

Finally, the complete description of the rule considering the target attribute and the weight would be:

IF Al is {Low, Medium} and Ca is {Low} and Mg $\lessapprox$ K and

SUBT (Na, Mg) is {VeryLow, Medium, High, VeryHigh} THEN

TypeOfGlass is tableware with weight 0.97

being all variables defined with five uniformly distributed labels within the range of their domains and also taking into account that label *"Low"* of a variable does not have to coincide with the same label of a different variable.

The previous rule would be easily interpreted as:

"IF Aluminium is Low or Medium and Calcium is Low and Magnesium

is approximately less than or equal to Potassium and Sodium minus Magnesium

is not Low THEN TypeOfGlass is Tableware with weight 0.97"

4.4.1 The genetic algorithm of NSLV-FR

Once the main ideas have been exposed, the next step is to describe the characteristics of the algorithm associated to the model previously explained, that is, NSLV-FR (NSLV Functions and Relations) [34]. Thus, some of its properties are listed below:

- It uses an additional level in the representation of a rule, either the relation or the function level, depending on the version it is being compared with (NSLV-F or NSLV-R).
- It keeps the same genetic operators as NSLV for variables, values and consequent, using two-point crossover and mutation operators for integer coding in the relation and function levels.
- The mutation operation defined for the function and relation levels uses a probability that establishes if the corresponding chromosome has been modified (in the experimental study this value has been set to **0.2**). When this occurs, a position of the level is randomly selected. If that position has zero value a new random value is generated for selecting a new function of the catalog. If the position has a different value from zero, then zero value is assigned.
- The two points crossover operator and the binary uniform mutation operator are applied over the new value level associated to the function level.
- All individuals from initial population start having the genes of the relation and function levels with zero value. That means that at the beginning of the genetic process no functions neither relations are considered.
- Before the evaluation of each rule, the algorithm uses a mechanism for removing repeated relations and functions. It considers only the first occurrence of each one of them from left to right.

One of the main disadvantages associated to the use of relations and functions is the increasing of the searching space, so that many rules are needed for covering few examples of a class (too specific rules).

For solving this issue, the completeness degree related to a rule plays an additional role in the fitness function. In this way, a rule will be considered only if it covers a certain percentage of the examples of the class the rule belongs to. So, the first component of the fitness function changes. On the one hand, it introduces the weight of the rule raised to the number of conditions in order to penalize those rules having a high number of conditions in the antecedent. On the other hand, the completeness degree is added as a condition together with a covering threshold for considering the rule. This threshold is iteratively decreased to make this condition more restrictive.

Thus, the fitness function for NSLV-FR can be summarized in the following expression:

$$fitness(R) = [\Psi_\delta(R),\ svar(R),\ sval(R)]$$

being:

- $\Psi_\delta(R)$: the first component of the fitness function, that works in the following way:

$$\Psi_\delta(R) = \begin{cases} (\Gamma'_{k_1,k_2}(R) \times \Lambda(R) \times w^n) & \text{if } (\Lambda(R) \geq \delta) \\ -\infty & \text{otherwise.} \end{cases} \tag{4.8}$$

- $\Gamma'_{k_1,k_2}(R)$: a modified version of $\Gamma_{k_1,k_2}(R)$, being the last one the original consistency degree detailed in Section 3.1.3.2.2. This modification has been included in order to consider those situations where the number of successes of a rule is less than the number of failures, being $\Gamma_{k_1,k_2}(R) > 0$. The new expression used to define the consistency degree is:

$$\Gamma'_\delta(R) = \begin{cases} \Gamma_{k_1,k_2}(R) & \text{if } (n_s(R) > n_f(R)) \\ 0 & \text{otherwise.} \end{cases}$$

  where $n_s(R)$ is the number of successes of the rule and $n_f(R)$ is the number of failures.

- $\Lambda(R)$: the completeness degree, whose expression is detailed in Chapter 3.1.3.2.2.

- w: the weight of the rule.

- n: the number of conditions, understanding as conditions the initial variables, relations and functions considered in the antecedent of a rule.

- $svar(R)$: the simplicity degree of a rule determined by the number of irrelevant variables. The expression is detailed in Section 3.2.2.3.3, equation 3.31.

- $sval(R)$: the simplicity degree of a rule determined by the number of understandable assignments to the variables of a rule. The expression is detailed in Section 3.2.2.3.3, equation 3.32.

- δ: the threshold that represents the minimum percentage of examples (of the class that is being learned), that must be covered by a rule. The initial value for δ has been experimentally fixed.

It is important to point out that the value w^n appears in the fitness function with the objective of obtaining simpler rules, as it acts limiting the number of conditions appearing in those rules.

It is also important to note that the selection of the best rule (through the fitness function), keeps being guided by a lexicographical order, as occurred with previous versions.

4.5 Experimental study

In this section, the aforementioned versions, NSLV-R, NSLV-F and NSLV-FR, are compared with each other and also with the base algorithm NSLV. The statistical tests applied in the study are the Friedman and Iman-Davenport tests and the post-hoc procedures of Holm and Shaffer. The information about the databases used in the comparison as well as the rest of assumptions are available in Chapter A. Some extra details will be given below regarding to the set of pre-defined relations and functions considered in the study.

4.5.1 Pre-defined relations

The set of relations considered by the learning algorithm consist in four fuzzy relations and two crisp ones:

- Approximately equal to
- Very different to
- Approximately less than or equal to
- Approximately greater than or equal to
- Less than
- Greater than

All these relations are defined for variables X_i and X_j with bounded domains $[inf_i, sup_i]$ and $[inf_j, sup_j]$ respectively and verifying that $degree_o(X_i, X_j) > 0.9$.

Associated to the domains of these variables, we define the parameter

$$q_{ij} = \frac{|I_i - I_j|}{c}$$

with $I_i = min\{sup_i, sup_j\}$, $I_j = max\{inf_i, inf_j\}$ and where c is a strictly positive parameter (the experiments were performed using $c = 10$). The value q_{ij} represents the size of the partition in c fragments of the domain of the difference variable $|X_i - X_j|$. Thus, let us describe each one of the relations considered:

The *approximately equal to* relation:

$$\mu_{X_i \approx X_j}(x_i, x_j) = \begin{cases} 1 - |\frac{x_i - x_j}{q_{ij}}| & \text{if } |x_i - x_j| < q_{ij} \\ 0 & \text{otherwise.} \end{cases}$$

The *very different to* relation:

$$\mu_{X_i<>X_j}(x_i, x_j) = \begin{cases} 0 & \text{if } |x_i - x_j| < (c-1)q_{ij} \\ \frac{|x_i - x_j| - (c-1)q_{ij}}{q_{ij}} & \text{otherwise.} \end{cases}$$

The *approximately less than or equal to* relation:

$$\mu_{X_i<\approx X_j}(x_i, x_j) = \begin{cases} 1 & \text{if } x_i < x_j \\ 1 - |\frac{x_i - x_j}{q_{ij}}| & \text{if } 0 < |x_i - x_j| < q_{ij} \\ 0 & \text{otherwise.} \end{cases}$$

The *approximately greater than or equal to* relation is included in an indirect way since $X_i >\approx X_j$ is equivalent to $X_j <\approx X_i$.

4.5.2 Pre-defined functions

Regarding to the set of functions taken into account in the experimental study, they can be summarized in:

- SUM (X_i,X_j), defined as $X_i + X_j$

  Restriction: $Unit(X_i) \neq 0$ and $Unit(X_j) \neq 0$ and $Unit(X_i) = Unit(X_j)$.

- SUBT (X_i,X_j), defined as $X_i - X_j$

  Restriction: $Unit(X_i) \neq 0$ and $Unit(X_j) \neq 0$ and $Unit(X_i) = Unit(X_j)$.

- PRODUCT (X_i,X_j), defined as X_i * X_j

  Restriction: $Unit(X_i) \neq 0$ and $Unit(X_j) \neq 0$.

- DIV (X_i, X_j), defined as $\frac{X_i}{X_j}$

  Restriction: $Unit(X_i) \neq 0$ and $Unit(X_j) \neq 0$

being $Unit(X)$ the measuring unit associated to variable X. If $Unit(X) = 0$, it means that X can not be combined with any other variable. Other different value ($Unit(X) \neq 0$), means that the involved variable can be combined with those that have the same value as it has.

The **SUM** and **SUBT** functions can only be applied over variables with the same measuring units, while **PRODUCT** and **DIV** operations can be applied over variables with different measuring units. The fact is that in the KEEL platform the databases do not consider units, so in order to develop this study we have assumed that for each database all variables can be combined. It is important to remark that the domain of the new variables generated will also be divided into five uniformly distributed linguistic labels.

Table 4.1 show the configuration of the different parameters associated to the algorithms involved in the comparison.

Table 4.1: Specific conditions for NSLV, NSLV-R, NSLV-F and NSLV-FR. NAV means the number of antecedent variables and *n_class* represents the number of classes of the specific problem.

	Specific Conditions
Size of genetic population	20*n_class
Number of iterations	500
Mutation prob. (Value level)	0.01
Mutation prob. (Variable level)	1/NAV
Mutation prob. (Consequent level)	0.01
Crossover prob.	1

This analysis will be centred in the study of four parameters: the accuracy on training and testing sets, the average number of rules and the time employed to get the model.

Table 4.2: Average Rankings of the algorithms (Friedman) and computed p-values by Friedman and Iman-Davenport (accuracy on training set).

Algorithm	Ranking
NSLV	3.15
NSLVR	2.7875
NSLVF	2.375
NSLVFR	1.6875
Friedman p-value	Iman-Davenport p-value
0.000003	0.000000618749

According to the results shown in tables 4.2 and 4.3, we can see that NSLV-FR obtains statistical differences with regard to the rest of algorithms involved in the comparison, that is, NSLV-R, NSLV-F and NSLV.

Table 4.3: Adjusted p-values (accuracy on training set).

i	algorithm	unadjusted p	p_{Holm}
1	NSLV	0	0.000001
2	NSLVR	0.000139	0.000277
3	NSLVF	0.017239	0.017239

Looking at tables 4.4 and 4.5, we observe that NSLV-F is the best algorithm in ranking and obtains significant differences when compared with NSLV (attending to Holm's test). In order to check all possible pairwise comparisons, we also applied Shaffer's test. Paying attention to Table 4.6, we can see that all the algorithms working with feature construction obtain significant differences with respect to NSLV. So, joining with the results shown in tables B.5 and B.6, we can see that NSLV-FR achieves the best average results although it is the third one in ranking classification (talking about test parameter), being the best one in training. This means that the

Table 4.4: Average Rankings of the algorithms (Friedman) and computed p-values by Friedman and Iman-Davenport (accuracy on testing set).

Algorithm	Ranking
NSLV	3.15
NSLVR	2.3375
NSLVF	2.075
NSLVFR	2.4375
Friedman p-value	Iman-Davenport p-value
0.001652	0.001179981795

Table 4.5: Adjusted p-values (accuracy on testing set).

i	algorithm	unadjusted p	p_{Holm}
1	NSLV	0.000196	0.000588
2	NSLVFR	0.209211	0.418423
3	NSLVR	0.363178	0.418423

model including both, relations and functions, presents some overfitting. On the other hand, NSLV-F is the one having the best global behavior (regarding to training and test parameters), as it is the best algorithm in test and the second one in training (talking about ranking classification) and closely followed by NSLV-R in average values.

Table 4.6: Adjusted p-values (accuracy on testing set)

i	hypothesis	unadjusted p	p_{Shaf}
1	NSLV vs .NSLVF	0.000196	0.001177
2	NSLV vs .NSLVR	0.004884	0.014652
3	NSLV vs .NSLVFR	0.01358	0.040741
4	NSLVF vs .NSLVFR	0.209211	0.627634
5	NSLVR vs .NSLVF	0.363178	0.726355
6	NSLVR vs .NSLVFR	0.729034	0.729034

Now, focusing our attention in the number of rules (tables 4.7, 4.8 and B.7), we realize that NSLV presents the best results but closely followed by NSLV-R and NSLV-F (with very similar results in ranking). It obtains significant differences with NSLV-FR. Regarding to average values (Table B.7), NSLV achieves almost 5 rules less in average than the worst version, NSLV-FR, which means that the use of feature construction increases the complexity of the rule base.

Finally, with regard to the time parameter (tables 4.9 and 4.10), we observe that NSLV-R (and given the extremely close ranking value, NSLV), significantly win NSLV-F and NSLV-FR. Both pairs present similar values in ranking (NSLV with NSLV-R and NSLV-F with NSLV-FR), but quite different results when talking about average values (B.8). Even though

Table 4.7: Average Rankings of the algorithms (Friedman) and computed p-values by Friedman and Iman-Davenport (average number of rules).

Algorithm	Ranking
NSLV	2.3
NSLVR	2.3125
NSLVF	2.35
NSLVFR	3.0375
Friedman p-value	Iman-Davenport p-value
0.02582	0.023823766813

Table 4.8: Adjusted p-values (average number of rules).

i	algorithm	unadjusted p	p_{Holm}
1	NSLVFR	0.010626	0.031877
2	NSLVF	0.86249	1.72498
3	NSLVR	0.965461	1.72498

Table 4.9: Average Rankings of the algorithms (Friedman) and computed p-values by Friedman and Iman-Davenport (time employed to obtain the model).

Algorithm	Ranking
NSLV	1.775
NSLVR	1.7
NSLVF	3.325
NSLVFR	3.2
Friedman p-value	Iman-Davenport p-value
0	0

Table 4.10: Adjusted p-values (time employed to obtain the model).

i	algorithm	unadjusted p	p_{Holm}
1	NSLVF	0	0
2	NSLVFR	0	0
3	NSLV	0.795012	0.795012

Table 4.11: Results obtained by the algorithms in the 9^{th} partition of the *wdbc* database. The table shows the accuracy on training and testing sets.

Algorithm	Training (%)	Test (%)
NSLV	98.6	92.9
NSLV-R	99.2	98.2
NSLV-F	99	94.7
NSLV-FR	98.8	98.2

NSLV-R is the best algorithm according to the results given by the Friedman and Holm tests, it invests almost ten seconds more in average than NSLV. The same occurs between NSLV-F and NSLV-FR, where the last one is almost 25 seconds faster in average than the one using only functions.

Thus, considering the global results we can extract some conclusions. Talking in general terms and according to the ranking classification, we find that:

- NSLV-FR shows some overfitting, as it is the best algorithm in training, the third one in test and the model which obtains the rule base with higher number of rules.
- NSLV-F is the best model in accuracy. Anyway, its average results are rather similar to NSLV-R in relation to the number of rules. This means that the model using functions is more accurate than the one using relations. So, taking these facts into account, may be functions provide more information than relations. Nevertheless this could be a hasty assertion, as it would depend on the problem, the restrictions about the combinations between variables, etc.

In order to prove the influence of the feature construction approach over the extracted rules, we will now expose some examples of rule bases obtained (by the different algorithms), for the *wdbc* database (Figure A.1). Table 4.11 shows the accuracy of the different proposals when running the 9^{th} partition of the *wdbc* dataset.

According to the rulesets shown in figures 4.11, 4.12, 4.13, 4.14 and tables 4.11 and 4.12, we can see that in the aforementioned partition used in this example, the algorithms obtaining the higher accuracy are NSLV-R and NSLV-FR. Looking at Table 4.12, we observe that NSLV-R has also the rule base with higher number of rules while NSLV-FR has the lower value. This explains, in part, the high accuracy achieved (by NSLV-R) when compared with NSLV-F, for instance.

One more point with regard to NSLV-R is that all rules part of the rule base are used for classifying testing examples, which means that the rules learned from the training set are all useful in order to represent the testing set. On the other hand, NSLV-FR gets also the higher accuracy (together

Table 4.12: For each one of the algorithms, this table shows the number of successes/fails of each rule part of the different rule bases in the testing set.

	NSLV		NSLV-R		NSLV-F		NSLV-FR	
#Rule	Succ.	Fails	Succ.	Fails	Succ.	Fails	Succ.	Fails
R0	31	3	13	0	9	0	30	0
R1	12	1	2	0	11	0	16	0
R2	2	0	6	0	7	1	3	0
R3	0	0	2	1	0	2	0	1
R4	1	0	8	0	25	0	5	0
R5	3	0	2	0	0	0	2	0
R6	0	0	11	0	1	0	-	-
R7	3	0	1	0	1	0	-	-
R8	1	0	8	0	0	0	-	-
R9	-	-	2	0	-	-	-	-
R10	-	-	1	0	-	-	-	-

with NSLV-R), but employing less rules. In this case, the learned rules own more information themselves.

Now, if we pay attention to the number of successes/fails, we notice that considering the two more informative rules from each ruleset, one of them does use relations or functions (in the algorithms using feature construction, of course). So, feature construction methods could be seen as a way to refine the searching process. We also stand out that some of the most representative attributes appearing in the rules are *Area3* and *Concave_points1* (this last one is involved in some relations and functions), which enhances the importance of both of them when describing the set of examples.

4.6 Conclusions

In this chapter we have talked about *feature construction* in a fuzzy rule learning algorithm. This technique has been developed through two different approaches. The first one uses relations in the antecedent of the rules while the second one employs functions en the antecedent of such rules. Both of them pursue a common purpose: to extract additional information from the original variables.

On the one hand, the use of relations allows making flexible partitioning of the input space. This translates in models with a good trade-off among accuracy and interpretability, maintaining the simplicity in the description.

On the other hand, by using functions in the antecedent of fuzzy rules, it is possible to increase the number of variables describing the problem through the combination of the original ones.

Both methods employ a structure to store either the most promising relations and functions, called *Catalog of Relations* (CR) and *Catalog of*

Functions (CF). They act as an index set where each entry is a relevant relation or function to be considered during the learning process.

Thus, in the chapter, three algorithms have been presented. The first two algorithms introduce the use of relations and functions respectively (NSLV-R and NSLV-F), while the third one (NSLV-FR) combines both methods in order to exploit the main advantages of the relations and functions. As a consequence of the increasing of the searching space associated to the use of feature construction methods, NSLV-FR introduces a new criterion in the fitness function based in the use of a threshold to ensure a minimum percentage of examples covered by each rule.

The experimental results show that the use of relations and functions (either separated or combined methods), improves the prediction capability of the rule model extracted. As a consequence, the number of rules and the average time employed to obtain the model is also increased. Anyway, the significant differences achieved in accuracy in favour of the algorithms using feature construction techniques, justify a relative increase in the rest of parameters studied.

R0: IF Concave_points1 = { VeryLow Low} Area2 = { VeryLow} Area3 = { VeryLow} Compactness3 = { VeryLow Low} THEN Class IS B W 0.9279778959021685

R1: IF Perimeter3 = { Medium High VeryHigh} THEN Class IS M W 0.9427146601010222

R2: IF Smoothness1 = { Medium High VeryHigh} Compactness2 = { VeryLow Low High VeryHigh} Concavity2 = { VeryLow Low} Fractal_dimension2 = { VeryLow Low} Texture3 = { Medium High VeryHigh} Concave_points3 = { Medium High VeryHigh} THEN Class IS M W 0.8289296712126152

R3: IF Texture1 = { Medium VeryHigh} Symmetry1 = { Low} Compactness2 = { VeryLow Low High} Concavity2 = { VeryLow Low} Fractal_dimension2 = { VeryLow Low} Compactness3 = { Low} Concave_points3 = { Medium High VeryHigh} THEN Class IS M W 0.7284836582193538

R4: IF Texture1 = { VeryLow} Area1 = { VeryLow Low} Smoothness1 = { VeryLow Low VeryHigh} Perimeter2 = { VeryLow} Symmetry3 = { VeryLow Low} THEN Class IS B W 0.9877149519361487

R5: IF Texture1 = { Medium High VeryHigh} Concave_points2 = { Medium High VeryHigh} Area3 = { Low Medium VeryHigh} Concave_points3 = { VeryLow High VeryHigh} THEN Class IS M W 0.9624991605824593

R6: IF Radius1 = { Medium High} Texture1 = { High VeryHigh} Concave_points1 = { Low Medium High} Symmetry1 = { Medium High VeryHigh} Texture3 = { VeryLow Low High VeryHigh} THEN Class IS M W 0.9479497004775931

R7: IF Perimeter1 = { VeryLow Low} Compactness1 = { VeryLow Low} Symmetry1 = { VeryLow Low VeryHigh} Perimeter2 = { VeryLow High} Concave_points2 = { VeryLow Low} Texture3 = { VeryLow Low} Smoothness3 = { VeryLow Low VeryHigh} Concavity3 = { VeryLow Medium} Concave_points3 = { VeryLow Medium} Symmetry3 = { VeryLow Low} THEN Class IS B W 0.9761545147538883

R8: IF Concavity2 = { VeryLow Low} Symmetry3 = { High VeryHigh} THEN Class IS M W 1.0

Figure 4.11: Ruleset obtained by the algorithm NSLV in the 9^{th} partition of the wdbc database.

R0: IF Concave_points3 ⩽E Smoothness1 THEN Class IS B W 0.9789855072463768

R1: IF Perimeter1 = { Low} Symmetry3 = { VeryLow Low} Concave_points1 ⩽E Compactness2 THEN Class IS B W 0.9953421120333216

R2: IF Concave_points3 ! =E Compactness2 Radius1 ! =E Area2 Fractal_dimension1 ⩽E Concavity1 THEN Class IS M W 0.9682109452978712

R3: IF Radius3 = { VeryLow Low VeryHigh} Texture3 = { VeryLow Low VeryHigh} Concave_points3 = {VeryLow Low Medium} Symmetry3 = { Low} Concavity1 < Compactness1 THEN Class IS B W 0.9602658183702274

R4: IF Symmetry1 = { VeryLow Medium High VeryHigh} Symmetry2 = { Low Medium High VeryHigh} Symmetry3 = { VeryLow Low} Concavity3 < Symmetry3 Area2 ⩽E Texture3 Concave_points1 < Fractal_dimension1 THEN Class IS B W 0.9942367244275221

R5: IF Radius1 = { Medium High} Texture1 = { Medium High VeryHigh} Compactness2 = { VeryLow Low Medium} Area2 ! =E Radius3 THEN Class IS M W 0.9840801436017024

R6: IF Compactness2 = { VeryLow Low High} Texture3 = { VeryLow Low Medium} Concave_points1 ⩽E Compactness2 Concave_points1 < Fractal_dimension1 THEN Class IS B W 0.9955326739518494

R7: IF Radius3 = { Medium High VeryHigh} Texture3 = { Medium High VeryHigh} THEN Class IS M W 0.962907449522046

R8: IF Texture3 = { Medium High VeryHigh} Perimeter3 = { Medium High VeryHigh} Area2 ! =E Radius1 THEN Class IS M W 0.9853738371603674

R9: IF Symmetry1 = { Medium High VeryHigh} Texture3 = { Medium High VeryHigh} Compactness3 = { VeryLow Medium High VeryHigh} Fractal_dimension3 = { Low Medium} Smoothness3 < Concave_points3 THEN Class IS M W 0.962883542223363

R10: IF Concave_points1 = { VeryLow Medium High VeryHigh} THEN Class IS M W 0.3746933668901874

Figure 4.12: Ruleset obtained by the algorithm NSLV-R in the 9^{th} partition of the wdbc database.

R0: IF Area3 = { VeryLow} Concave_points3 = { VeryLow Low} THEN Class IS B W 0.9749258286226512
R1: IF (Concave_points1*Texture3) = { L1 L4} THEN Class IS M W 0.9961450989002424
R2: IF Symmetry3 = { Low Medium High VeryHigh} (Texture2+Radius3) = { L1 L2 L4} THEN Class IS M W 0.9496300316306346
R3: IF Concavity1 = { VeryLow High} Area3 = { VeryLow} Smoothness3 = { VeryLow Low Medium} THEN Class IS B W 0.9637171703782214
R4: IF Area3 = { VeryLow} Concavity3 = { VeryLow High VeryHigh} Symmetry3 = { Low} THEN Class IS B W 0.9783593589599837
R5: IF Texture3 = { High VeryHigh} Concave_points3 = { Medium High VeryHigh} THEN Class IS M W 0.9258864889435975
R6: IF Texture1 = { VeryLow} Concavity3 = { VeryLow Low} THEN Class IS B W 0.9753145946537688
R7: IF Fractal_dimension1 = { VeryLow} (Concave_points1-Fractal_dimension1) = { L0 L2 L3} THEN Class IS M W 0.9385971221300116
R8: IF (Smoothness2+Concave_points3) = { L0 L2 L3} THEN Class IS M W 0.3475388454968819

Figure 4.13: Ruleset obtained by the algorithm NSLV-F in the 9^{th} partition of the wdbc database.

R0: IF Area3 = { VeryLow} Concave_points3 = { VeryLow Low} THEN Class IS B W 0.9749258286226512
R1: IF (Radius1/Area3) = { L0 L2} Fractal_dimension1 < Concave_points1 Smoothness1 < Concavity1 THEN Class IS M W 0.9998451137419374
R2: IF Texture3 = { Medium High VeryHigh} (Radius1/Area3) = { L0} Concave_points3 ! =E Compactness2 Fractal_dimension1 ⩽E Concavity1 THEN Class IS M W 0.9732671722874883
R3: IF Compactness2 = { VeryLow Medium High VeryHigh} Radius3 = { Medium VeryHigh} Texture3 = { Medium High VeryHigh} THEN Class IS M W 0.9490180336074999
R4: IF Symmetry1 = { VeryLow Medium VeryHigh} Texture3 = { VeryLow Low} (Compactness1-Perimeter3) = { L1 L2 L3} THEN Class IS B W 0.8258497508137139
R5: IF (Concavity2-Radius3) = { L1 L2 L3 L4} (Radius3-Perimeter3) = { L1 L2 L3 L4} Concave_points2 ! =E Concave_points1 THEN Class IS M W 0.9387358486204522

Figure 4.14: Ruleset obtained by the algorithm NSLV-FR in the 9^{th} partition of the wdbc database.

4.7 An application of a feature construction algorithm to remotely sensed imagery

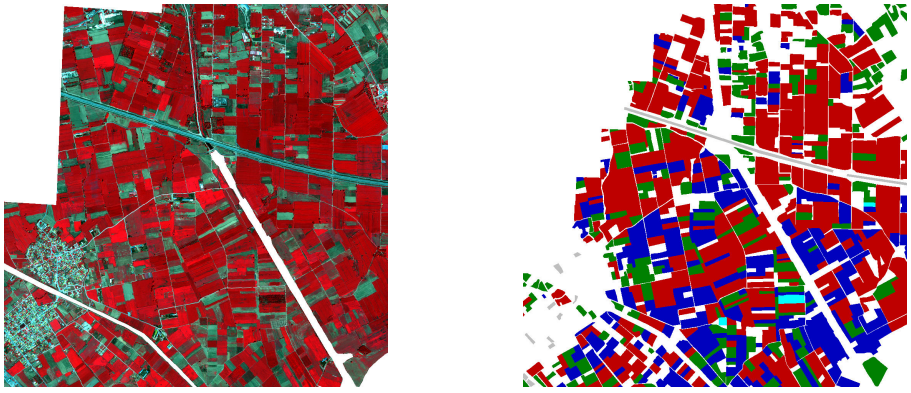
This section is the result of a three months stay in the Aristotle University of Thessaloniki. During this time the work in Thessaloniki was supervised by Professor J. B. Theocharis. Next subsections will describe the cooperative work developed as well as the results obtained.

4.7.1 Introduction

In certain real-world applications, some complex relations exist between the input variables (features). Such relations are typically identified through expert knowledge and after careful examination of the natural properties of the problem at hand. Thus, the learning algorithm NSLV-FR provides an useful tool to automatically infer more complex relationships among the input variables, through the feature construction technique.

Remote sensing classification tasks are typical examples of problems where the input variables are interdependent. The primary information is provided by satellite-borne (or airborne) sensors, which collect the earth's reflected solar radiation in specific portions of the electromagnetic spectrum. It is well-known that different land cover types (and especially plants), absorb specific regions of the spectrum and reflect the remaining radiation [115]. This property has been exploited in the past for devising a number of indices that characterize specific properties of vegetation (greenness, humidity, chlorophyll content, etc.) [94]. For example, the most frequently employed vegetation index is the so-called normalized difference vegetation index (NDVI), which is calculated as the normalized difference between the near-infrared (NIR) and red channels of the image. NDVI relies on the fact that vegetation absorbs the red portion of the spectrum for photosynthetic purposes, but reflects most of the NIR radiation. To this end, green vegetation can be easily differentiated from other land cover types, as it exhibits high NDVI values. This is the primary reason for which all modern sensors targeting at land cover applications bear a NIR channel.

Nevertheless, well-established vegetation indices typically characterize broad categories of vegetation types and not specific species, despite the advent of hyperspectral spectroscopy, which has led to the construction of many new vegetation and chlorophyll content indices [52]. Moreover, equivalent relations for higher-order spectral and textural features (commonly extracted in order to increase classification accuracy when using multispectral images), do not exist. So, the main idea here is to apply a GFRBCS that exploits a feature construction methodology on a crop classification task using multispectral satellite imagery. The objective is to identify any new useful relationships between the input variables that can identify the specific crop species of the study area, by analyzing the linguistic interpretation of



(a) (b)

(c)

Figure 4.15: Study area: (a) pseudo-color composite of the satellite image, (b) the labeled testing polygons, and (c) the respective legend.

the fuzzy rules achieved by the learning algorithm NSLV-AR.

4.7.2 Study area and dataset description

As study area for this work we have selected an agricultural region northwest of Lake Koronia, which is a lake-wetland ecosystem of ecological importance in northern Greece (40° 41′ N, 23° 09′ E). An IKONOS bundle image (exhibiting 4 m spatial resolution in the multispectral channel) was acquired on 7 August 2005 for classification purposes. The IKONOS image has three bands in the visible region of the electromagnetic spectrum, which will be hereafter referred as red (R), green (G) and blue (B) bands/features, and one in the near-infrared (NIR) region.

A portion of 1000×1000 pixels from the original image was used in this paper. Figure 4.15(a) depicts a false-color composite of the satellite image, using the NIR, R, and G bands in lieu of the red, green, and blue channels of the color image, respectively. This specific view exploits the vegetation properties discussed in the introduction: green vegetation absorbs visible radiation and reflects NIR; hence it appears red in the color image (only the first channel has high values). Other land cover types also reflect a portion of the visible radiation and so they exhibit different colorings.

The class scheme includes five classes: alfalfa, cereals (primarily wheat), maize, orchards, and artificial surfaces (roads and buildings). After extensive field surveys and subsequent careful photo-interpretation, a large number of polygons were identified in the image and assigned to one of the five classes

(Figure 4.15(b)). All labeled pixels (577,316 in total) inside those polygons participate in the testing set. Moreover, 3000 random points (pixels) were selected for the training set. The training patterns were sampled from additional polygons, which do not participate in the testing set.

Due to the limited information provided by the four bands of the initial image, it is customary practice to extract additional higher-order features when dealing with classification tasks involving multispectral images. For this purpose, the following higher-order features were derived from the original bands of the image:

- *Spectral transformations.* We used the Brightness, Greenness, and Wetness components of the so-called tasseled cap transformation [58], as well as the first three components of the intensity-hue-saturation (IHS) transformation [120] (the saturation component exhibited almost zero variability in the whole image and was therefore not used).

- *Textural transformation.* As a source of textural information, we applied the gray-level co-occurrence matrix (GLCM) transformation [53]. For each band of the image, four measures from the co-occurrence matrix have been calculated: contrast (CON), angular second moment (ASM), correlation (COR), and homogeneity (HOM). For convenience, we use the notation <*measure*>:<*band*> in the following. For example, ASM:R denotes the angular second moment measure calculated from the red band of the original image.

A more thorough description of the study area and the extracted features can be found in [108]. The final feature space of our classification task comprises all 25 aforementioned features (4 initial bands, 5 transformed spectral features, and 16 GLCM measures). Of course, each fuzzy rule in NSLV-FR can also use any derivative relation and function feature from the respective catalog, as described in the previous section.

4.7.3 Experimental study

This experimental section aims to show how the interpretability properties of FRBCSs can be exploited in the remote sensing framework and (most importantly), how the feature construction abilities of NSLV-FR can identify new useful relationships in the specific land cover task considered.

Nevertheless, we first provide a brief comparative from the classification accuracy point of view, in order to validate whether NSLV-FR can achieve comparable performance with other classifiers. With this purpose, we consider two reference classifiers that have been shown to achieve very high accuracy in many and diverse classification tasks, namely, the support vector machine (SVM) and the random forest (RF) [13] classifiers. The parameters C and γ for the SVM classifier (we employed a radial basis function kernel)

Classifier	Classification accuracy (%)			
	Training		Testing	
	Average	Best	Average	Best
NSLV-FR	76.0	75.3	77.6	79.2
RF	100.0	100.0	82.6	82.8
SVM	84.8		82.4	

Table 4.13: Training and testing classification accuracies for the compared classifiers. For NSLV-FR and RF both average and best accuracies are reported.

were determined through a 5-fold cross-validation search in the training set, considering a grid of possible values. The final classification model was produced after retraining the SVM with the whole training set and the selected parameters. The same procedure was employed in order to determine the optimal number of trees and random features per tree for the RF classifier. Since NSLV-FR and RF exhibit stochastic characteristics, 20 independent runs were performed using different seeds for the pseudo-random generator.

Table 4.13 reports the comparative results for the three classifiers. For NSLV-FR and RF both average accuracies are reported, along with the exhibited by the best run, which is defined as the one achieving the highest accuracy on the testing set. Since SVM's learning algorithm is a deterministic one, only the respective accuracies for the (single) model are reported.

RF exhibits the highest testing accuracy, closely followed by SVM. NSLV-FR displays somewhat lower testing accuracy, although the absolute difference is not that big, especially if we consider the best model. In this analysis we should also take into consideration the much higher complexity of the former two classifiers. Indeed, NSLV-FR creates classification models with 40.6 rules on average, whereas each fuzzy rules comprises an average of 1.67 features. In contrast, the SVM model uses all features and comprises 1436 support vectors, whereas the RF model constructs 100 trees, each of which considers 10 random features. To this end, NSLV-FR's classification accuracy can be relatively considered as satisfactory.

In order to exploit NSLV-FR interpretability and feature construction properties, we focus on the best model obtained. As shown in Table 4.13, this model exhibits a testing accuracy of 79.2% (run 17). The primary source of misclassifications stems from the confusion between the alfalfa and maize classes, which are the major green vegetation species in our study area. Errors are also observed between the alfalfa and cereals classes, although to a relatively smaller extent. The latter case is observed because the image was acquired in August, when the cereals crops (most of which were wheat), had been harvested. Hence, sparsely vegetated alfalfa areas are confused with harvested cereal fields, because of the exposed soil. On the other hand, artificial structures can be easily discerned from all other classes, whereas

orchards represent a very small percentage of the total land cover.

Therefore, it would be very interesting to analyze the best NSLV-FR model in terms of the features constructed for the discrimination of the alfalfa and maize classes. For this purpose, the fuzzy rules for each class have been sorted in decreasing order with respect to their significance. As a measure of significance for each rule we have employed the (crisp) proportion of positive examples, that is, the number of training patterns that activate the rule and belong to the class described in its consequent, divided by the total number of this class's patterns.

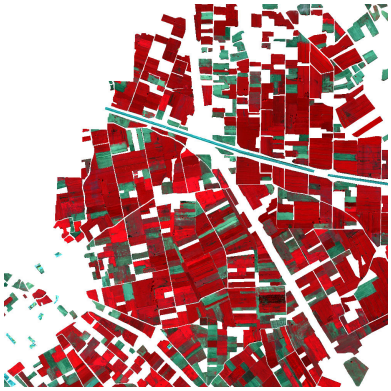
Here we present an example for the maize class. Each feature variable has been uniformly partitioned into fuzzy sets, for which we assign the linguistic labels {VerySmall, Small, Medium, Large, VeryLarge}. The antecedent part of the most significant fuzzy rule comprises two derivative function features:

$$\begin{aligned} &\textbf{IF } (blue/green) \text{ is } Large \text{ and} \\ &\quad (brightness + wetness) \text{ is } VerySmall \end{aligned}$$

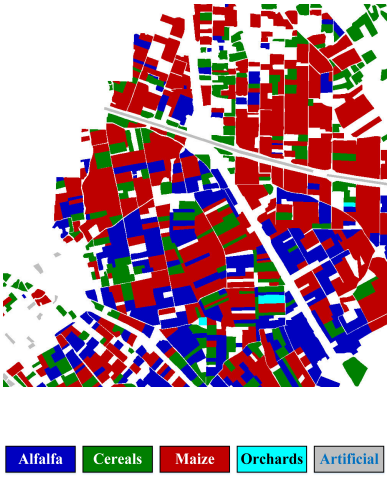
The rule uses a ratio between bands (blue and green) and the sum of two tasseled cap features (brightness and wetness). We should note that neither of these features have some well-established use or expert interpretation in the vegetation indices literature.

Figures 4.16(c) and 4.16(d), respectively, depict the grayscale representation of the two new features. For convenience, the study area's pseudo-color composite and the testing polygons have been replicated as the two top subfigures. Areas not belonging to the (testing) reference have been masked out from all images, in order to assist the visual comparison. The fuzzy rule declares that maize exhibits a high blue/green ratio. Indeed, all maize fields in Figure 4.16(c) have higher grayscale values (brighter areas). At the same time, however, maize is characterized by low values for the brightness+wetness feature (comparatively darker regions in Figure 4.16(d)). Since NSLV-FR employs the minimum operator for the conjunction of the individual predictive variables, both of these conditions must be satisfied in order for a pattern (pixel in our case) to be characterized as maize.

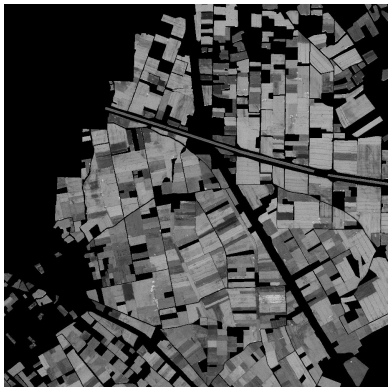
One way to visualize both features at the same time is to insert them into two channels of a color image and then try to interpret the different colorings produced, based on the description provided by the fuzzy rule. Such false-color composites are very popular in the remote sensing community (in the previous subsection we have actually provided such an example with the description of Figure 4.15(a)). In our case, however, an even easier way to produce an aggregate image is to just calculate the rule's activation degrees (memberships) for all image pixels. The result is shown in Figure 4.16(e). Comparing the latter representation with the reference sites (Figure 4.16(b)), it becomes apparent that the rule identifies almost all maize



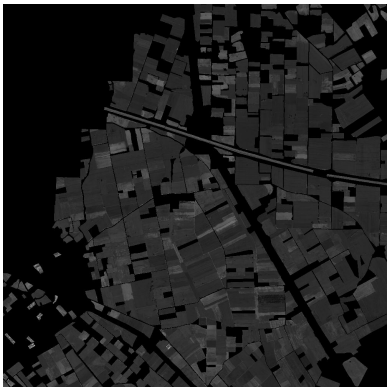
(a) Satellite image pseudo-color composite



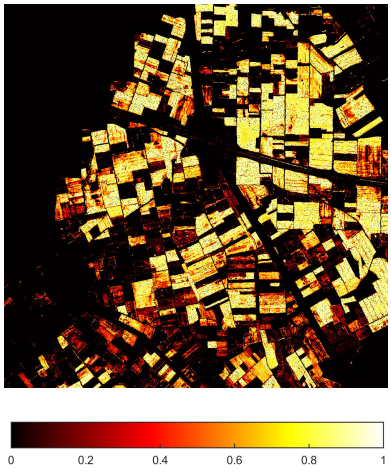
(b) Testing polygons



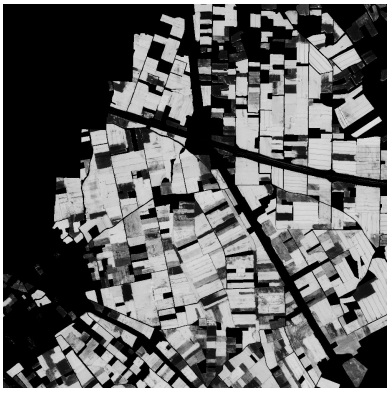
(c) (Blue/Green) feature



(d) (Brightness+Wetness) feature



(e) Rule activation degrees



(f) NDVI of the original image

Figure 4.16: The extracted features used by the most significant rule produced for the maize class and the rule's activation degrees for all pixels.

fields accurately. Lower (but non-zero) memberships are also observed for a few alfalfa fields as well. This is expected, since the two classes exhibit a strong correlation and are the main source of the misclassifications in the testing set. Nevertheless, most of these regions exhibit membership degrees below 0.5 and as such they can be easily identified.

If we compare the rule activation image and the pseudo-color composite of Figure 4.16(a) with the reference image, it is evident that the former is far more enlightening with respect to the maize/alfalfa discrimination. Although some alfalfa fields can be singled out from Figure 4.16(a) as vibrant red, most maize and alfalfa fields are rather indiscriminable. In the previous section we explained that the pseudo-color composite shown in Figure 4.16(a) relies on the absorption properties of green vegetation in the red and NIR portion of the electromagnetic spectrum. As mentioned in the introduction, NDVI is based on the same principle, being defined as $(\text{NIR} - \text{Red})/(\text{NIR} + \text{Red})$, and is ubiquitously employed to identify green vegetation (Figure 4.16(f)). Again, many alfalfa fields exhibit similar NDVI values with the typical maize ones and cannot be discriminated. To this end, it seems that NSLV-FR's feature construction ability can prove invaluable for easily discriminating related land cover types in a specific region.

4.7.4 Conclusions

So, in this section it has been presented the first results from the application of the feature construction concept in the context of land cover classification tasks from remotely sensed imagery. With this purpose, we exploited the advantages of the NSLV-FR classifier, which effectively embeds a feature construction procedure within a powerful genetic fuzzy rule-base learning framework. The experimental analysis was performed on a challenging crop classification task, using a very high resolution satellite image. The linguistic interpretation of the fuzzy rule base resulted in interesting conclusions, since we have been able to visually discriminate spectrally similar vegetation species.

Chapter 5

A modified version of the IRL approach to simplify the extracted knowledge

5.1 Motivation

In previous chapters, the influence of the amount of information in the learning process was addressed. As a consequence, one of the most important issues when working with genetic fuzzy rule learning algorithms is to obtain a good balance among accuracy and interpretability, even more in real-world problems [108, 40, 85, 7, 39, 93].

The feature construction methods are more focused in improving the accuracy of the model, because of the possibility of obtaining new information from the combination of the original variables. However, one of the side effects of this technique is related to its trend to obtain models with a high number of rules, which directly affects to the idea of interpretability.

In this chapter, the problem of the interpretability of the learned model is studied. In [32], it is shown a categorization of the interpretability measures based on two factors:

- Complexity versus semantic interpretability.
- Rule base versus fuzzy partition.

The taxonomy proposed in [32] comes from the combination of both factors. The scope of this study centres in the combination of complexity-based interpretability at the rule base level. Basically, interpretability in this case is related with the measure of the number of conditions and the number of rules of the fuzzy rule base.

So, when dealing with this topic, there are two main tasks in order to achieve interpretable rule models. One of them is related to the reduction of the number of conditions in a rule and the other one has to do with the reduction of the number of rules of the rule base.

In the literature it is possible to find many examples of complexity reduction of rule bases. Such is the case of [9], where Azam et al. propose a method to reduce the number of rules that are fired, keeping the performance of that of large rules, using a membership value of the linguistic variable to calculate a equilibrium value. Thus, rules are fired only if the inputs have membership value higher than the equilibrium value.

In [121], Zhao et al. propose a hybrid learning method combining a modified harmony search method with a fast recursive algorithm for learning a TSK-type rule-based fuzzy system with a compact model structure and a high model accuracy. With this purpose, they consider not only the number of fuzzy rules but also the structure of each rule premise and consequent. Thus, each fuzzy rule is associated with two sets of input attributes, in which the first is used for constructing the rule premises while the other is employed in the rule consequents.

Some other works can be found like [33], where a post-processing methodology to reduce the complexity of data-driven linguistic fuzzy models is proposed. The approach is based on rule selection via the formulation of a bi-objective problem with one objective focusing on accuracy and the other one on interpretability.

On the other hand, Soua et al. talk in [107] about improving the comprehensibility through a supervised learning method by automatic generation of fuzzy classification rules (SIFCO-PAF). The method reduces the complexity by decreasing the number of rules and of antecedent conditions, making it thus adapted to the representation and the prediction of rather high-dimensional pattern classification problems.

Finally, in [27] Fazzolari et al. investigate the influence of the application of a single granularity learning approach to the performance of fuzzy associative rule-based classifiers with the objective of reducing the complexity of the models obtained, trying to maintain a good accuracy on classification.

Next section addresses the main steps performed in order to give some solutions to the problem exposed.

5.2 Knowledge review

In Chapter 2, the IRL approach was described as the combination of the SC strategy together with GAs. This scheme characterizes by the iterative extraction of a single rule, which is the best one representing a set of examples of a class. Then, the rule becomes part of the final rule base.

It is easy that, during the learning process, some interesting rules improving in some way another one previously learned appear. Until now, each learned rule was irrevocably added to the rule base and no any other action could be carried out (at a rule base level). In order to consider this situation in which a new learned rule could improve some other and also to allow some action that handles the learning process in consequence, the *knowledge review* could be seen as a possible solution.

Thus, the knowledge review could be understood as a continuous evaluation process of the learned rules, that is, a refinement mechanism embedded in the learning process.

Including the knowledge review in an IRL approach implies three main tasks: learning a new rule, compare the new rule against the previously learned ones and to make a decision, which will be related to the replacement of one or more rules involved in the comparison. So now, the idea is to be able not only to add rules, but also to replace those which could be improved by the new learned one if appropriate.

Taking into account this philosophy, it will be next presented a first approximation of a IRL approach able to review the knowledge.

5.2.1 A sequential covering strategy for reviewing the knowledge

As stated before, the new capability of adding and replacing rules if considered, implies two different paths. Normally, by adding rules, the accuracy is increased (normal behavior of the IRL approach) and the replacement action is more related to the improvement of the interpretability. So, two different "objectives" are shown. The difficulty is to establish a procedure for comparing both terms. In this work, a two-rules extraction process in order to measure both parameters is presented. This means that after each iteration, two different rules are extracted: the best one increasing the accuracy and the best one improving the interpretability.

Thus, following the considerations above, in [36], an iterative model able to replace rules was proposed. It was based on the learning algorithm NSLV. As it has been just said, the main difference now is that in each step a decision is taken among adding the best rule increasing the accuracy of the rule base, adding the best rule replacing one or more rules previously learned or adding no rules if it is considered that the rule base can not be improved. So, the sequential covering presents the following structure:

SEQUENTIAL_COVERING (E, f)

1. Learned_rules ← {}
2. Action ← RETURN_ONE_ACTION (E, f, Learned_rules, new_rule, rules_to_remove)
3. **While** (Action ≠ STOP) **Do**
   (a) Learned_rules ← Add (Learned_rules, new_rule)
   (b) **If** (Action == REPLACE) **Then**
      i. Learned_rules ← Remove (Learned_rules, rules_to_remove)
   (c) E ← Penalize (Learned_rules, E)
   (d) Action ← RETURN_ONE_ACTION (E, f, Learned_rules, new_rule, rules_to_remove)
4. **Return** Learned_rules

where the most important elements are described below:

- *RETURN_ONE_ACTION*: It is a procedure implemented through a genetic algorithm. It introduces some additional capabilities regarding to the classical LEARN_ONE_RULE procedure, in order to choose among the two possible rules extracted during the learning process, that is, the best rule increasing the accuracy or the best rule replacing rules. So, to select either of these options, this procedure must consider a decision criterion (later explained), returning not only the selected rule (new_rule) and the list of replaceable rules if appropriate (rules_to_remove) but also the action to be applied. It also incorporates a new genetic operator that replaces to the crossover operator in the following sense. The new operator applies the crossover operator as usual with probability p, and replaces the worst two individuals by two random rules from the set of learned rules, with probability $(1-p)$. In the experimental study has been considered $p = 0.05$.

- *Action*: Represents the selected action in the RETURN_ONE_ACTION procedure. It may take three possible values: "AGGREGATE" if the selected rule is the best that increases the prediction capability of the rule base, "REPLACE" if the chosen one is the best rule replacing rules and "STOP" if none of them is selected as the rule base can not be improved.

- *Add*: It is a function that adds the new rule to the set of learned rules.

- *Remove*: It is a function that removes the list of rules replaced by the new rule from the set of learned rules.

Regarding to the behavior of the algorithm, the first time the SC is run, the action carried out is to add the already learned rule. After that, while

more rules can be added to the rule base (that is, the rule base is able to go on being improved from the "interpretability" point of view), two different rules are learned in each iteration (the best rule increasing the accuracy and the best one replacing rules), but one of three possible actions may be taken. The first one is to "AGGREGATE" the best rule increasing the accuracy of the rule base. The second one is to "REPLACE" one or more rules of the rule base with the new one learned for this purpose (not increasing the accuracy, but maintaining and improving the interpretability). Finally, the last action ("STOP") implies that no rule is added. In this case, it is considered that no more rules are needed as the rule base cannot be improved. If the action carried out is "REPLACE", the rule base needs to be updated, which means that those rules that will be replaced by the new learned one, must be removed from the rule base.

Using this scheme, in which some learned rules can be replaced, the algorithm tries to reduce the complexity of the rule base (less rules), improving the interpretability. In order to allow this operation, one more consideration was taken into account. In the basic version, NSLV, one subpopulation for each class was used while this proposal maintains an extra subpopulation for rules able to replace rules. So, at the end of each call to the GA, two rules are available for choosing.

Thus, the evaluation function also introduces some changes for supporting this functionality. Now, the number of replaceable rules becomes part of the fitness:

$$fitness(R) = [\Psi(R),\ |\Omega(R)|,\ svar(R),\ sval(R)]$$

where:

- $\Psi(R)$ is the product of the consistency and completeness conditions.
- $|\Omega(R)|$ is the number of rules that can be replaced by R in the set of learned rules.
- $svar(R)$ the number of irrelevant variables.
- $sval(R)$ the number of understandable assignments.

In this fitness function all the elements are known but $|\Omega(R)|$, where $\Omega(R)$ represents the set of replaceable rules.

Thus, before going in this definition it is necessary to previously establish when a rule can replace a set of rules and to determine a criterion to update the rule base. In order to give a formal definition it is necessary to decide when a rule is critical for an example. This concept is associated to the SC strategy and uses the sequence of rules that have been learned at a certain time. The idea is that a rule is critical for an example when it is fired for this example, provides the correct class and if the alternative rule (the next

rule applicable for this example if any) is used, then it provides an erroneous classification of this example.

Definition 5.2.1 *Let E be an example set and Rules a set of already learned rules. We say that $R \in Rules$, is a* **critical rule for the example** $e \in E$ **given** *Rules iff:*

$$R \in FRules(e) \wedge$$

$$class(e) = consequent(R) \wedge$$

$$[|FRules(e)| = 1 \vee class(e) \neq consequent(Next(R, FRules(e)))]$$

where $FRules(e) \subseteq Rules$ is the set of rules with non zero adaptation to the example e sorted by the inference criterion and $Next(R, FRules(e))$ is a function that returns the rule that would be fired if R were not member of $FRules(e)$.

In order to clarify this definition, let us consider the next example. The rule R_4 is critical for the example e in these two possible situations:

- $FRules(e) = \{R_4\}$ is the set of firing rules (consisting only in one rule) and the class of e is equal to the consequent of R_4 or
- $FRules(e) = \{R_4, R_1, R_6, R_3\}$ is the set of firing rules in the order in which they would be fired for the example e, (R_4 before R_1 ... before R_3). In this situation, the class of e is equal to the consequent of R_4 and the next rule that would be fired (excluding R_4), that is, R_1, belongs to a different class from e. In other words, if the consequent of R_1 would be the same as R_4 (and equal to the class of e), then R_4 would not be a critical rule for e.

The critical rule for an example e given a set of rules *Rules* could not exist, but if it exists, we can say that it is the only one critical rule for e. So, we can note it as $CriticRule(e,Rules)$ being able to take two possible values:

(1) $CriticRule(e,Rules)=R$, in this case, if R were eliminated of *Rules* the example e will be incorrectly classified,

(2) $CriticRule(e,Rules)=\emptyset$, in this case, the critical rule doesn't exist. This may happen for two reasons:

   a) e is not correctly classified by *Rules* or

   b) two o more rules of *Rules* correctly classify the example e.

Thereby, given an example set E and a rule set *Rules* and applying the previous definitions we can determine if there are rules in *Rules* that could be removed from it, keeping the accuracy of the modified rule base. Only the rules in *Rules* that are critical rules of at least one example are needed

for keeping the classification capacity. Following this idea, the definition of the set of replaceable rules that is produced when a new rule is added to the set *Rules* is given below.

Definition 5.2.2 *Let E be a set of examples, Rules a set of learned rules and R a new rule, $R \notin Rules$. We define the set of rules from Rules that can be removed when R is added to Rules as:*

$$\Omega(R) = \{R' \in Rules \mid \nexists\, e \in E \text{ such as } R' = CriticRule(e, Rules \cup R)\}$$

Coming back to the fitness definition, one more time the fitness function follows a lexicographical order in which the first component dominates in the selection of the best rule and the rest of them are used if a tie situation occurs.

As it was previously said, when the RETURN_ONE_ACTION procedure was described, it is necessary to use a decision criterion to know which one among the two options (rules) to select in order to update the rule base. Since these two "best rules" have different "objectives", it is not possible to measure or compare them from a quantitative point of view. So, the decision can not be based in which one of them is better. Thus, the choice is based in a logic criterion taking into account that the priority now is to improve the interpretability at a rule base level. In this sense, if the rule that increases the accuracy of the model has a number of examples correctly classified greater than the number of replaceable rules of the best rule replacing rules, then the first one is selected. If not, the second one is chosen. We prefer reducing the number of rules of the rule base instead of considering a rule whose number of examples correctly classified is smaller than the number of replaceable rules.

So, summarizing, the knowledge review could be seen like a refinement process embedded in the learning process, since in the firsts stages the system tends to include new rules while in later stages the tendency is to replace rules and therefore to refine the knowledge seeking to improve its interpretability.

5.3 Main problems with knowledge review

In the preceding sections, it was mentioned that a rule replacing one or more rules is usually more general than those which are going to be replaced. This is important, because early replacements, may easily condition the rest of the rule extraction as the algorithm will converge sooner to more specific rules, being the first ones very difficult to replace later on.

As a consequence, the use of very specific rules increases the size of the rule base, so also the complexity. The point is that these specific rules usually cover few examples and they are almost useless in test set.

Thereby, next section is devoted to describe a possible solution in order to deal with these situations.

5.4 Pruning of searching spaces

As stated before regarding early replacements and also because of the normal behavior of the IRL approach, we are interested in finding a way to control the specificity level of the rules, since it has been demonstrated that too specific rules result useless in test set.

Some techniques were tried in order to solve this effect. One of them consisted in delaying the replacement of a rule forcing this operation only in final stages of the learning process. Rules were added to the rule base till the moment in which the replacement was allowed (something like a supervised learning). By using this strategy, no many rules were replaced and the complexity of the rule base was barely reduced.

Another technique was attempted using information about the clusters of each database. This information allowed to know the distribution of the examples in the databases as a preprocessing tool. Using these data to drive the learning process, the rule base was formed by significant but too general rules, which reduced the difference between training and test results in exchange for prediction capability.

Finally, the use of pruned searching spaces [35] gave us a tool which allowed to control 'how general' and 'how specific' could be the extracted rules. It demonstrated a good behavior, reducing the specificity level of the extracted rules (mainly in final stages). Thus, the idea now is related to define some restrictions on the completeness condition.

5.4.1 A SC strategy for searching on spaces pruned by a completeness condition

According to the explanations above and taking into account that the reducing of the searching space is seek, the objective is to modify the fitness function to avoid those rules covering few examples. This is done by rejecting the rules which are not supported by a minimum number of examples, through a parameter in the fitness.

Thereby, a parametric version of the fitness function is proposed:

$$\Psi_\delta(R) = \begin{cases} \Psi(R) & \text{if } \Lambda(R) \geq \delta \\ -\infty & \text{otherwise} \end{cases}$$

where δ is a threshold that represents the minimum percentage of positive examples that must be covered by a rule.

This expression was similarly used in Section 4.4.1, equation 6.1, when talking about feature construction. Now, the idea keeps being the same, but

without considering the parameter w^n, since the antecedent of the rules are not as complex as those including relations and functions.

So, the searching space is reduced as the algorithm only selects rules with a minimum completeness degree δ. In this sense, we say that the algorithm is working in a pruned searching space and the completeness conditions is determined by the parameter δ.

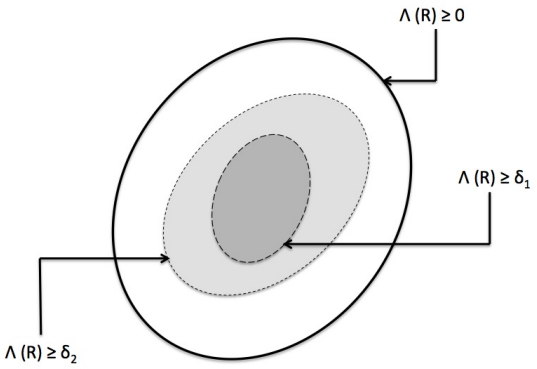


Figure 5.1: Pruned searching space in a decreasing sequence of the parameter δ. The completeness condition ($\Lambda(R)$) is restricted by this parameter.

The final expression of the fitness function is:

$$fitness(R, \delta) = [\Psi_\delta(R),\ svar(R),\ sval(R)].$$

The parameter δ determines a threshold representing the minimum number of examples that should be covered by a rule in order to be considered. For higher values of δ, the algorithm explores over rules with high redundancy in the training set. These rules tend to be more general (with few conditions in the antecedent), and represent the most relevant information in the example set. On the other hand, lower values of δ implies to collect less redundant information and, as a consequence, to obtain more specific rules.

Thus, the purpose is to use a decreasing sequence of δ so that the algorithm starts learning more general rules and in final stages learns more specific ones, but controlling the specificity level (Figure 5.1).

The decreasing sequence consists of k values of the parameter δ

$$\Delta = \{\delta_1, \delta_2, \ldots, \delta_k\}$$

where $\delta_i \in [0, 1]$ and $\delta_i > \delta_{i+1}$.

With this sequence a new sequential covering strategy is proposed. The main idea is to use the function $fitness(R, \delta)$ in a iterative scheme in which the parameter δ is moving in the previously defined order.

SEQUENTIAL-COVERING (E,F_δ,Δ)

1. Learned-rules ← {}
2. **For** i=1 to |Δ| **Do**
   (a) Rule ← LEARN-ONE-RULE (E,F_{δ_i})
   (b) **While** PERFORMANCE (Rule, E) > 0, **Do**
      i. Learned-rules ← Learned-rules + Rule
      ii. E ← Penalize{E}
      iii. Rule ← LEARN-ONE-RULE (E,F_{δ_i})
      iv. Learned-rules ← FILTER(Learned-rules)
3. **Return** Learned-rules

where E is the set of examples, F_δ the fitness function that depends on the δ parameter, Δ is a decreasing sequence of δ values and FILTER is a procedure to remove the rules already included in the set of rules *Learned-rules* and that has not been fired (using the inference process) by any example of E.

It was experimentally observed that the appropriate δ value differed for each dataset and it was not easy to estimate. Anyway, in [35], this experimental experience proved that the best values in most cases were in the interval [0.3, 0.1], choosing $k = 2$. In the configuration of the experimental study we have considered $\delta_{max} = \delta_1 = 0.3$ and $\delta_{min} = \delta_2 = 0.1$, for all datasets.

5.5 NSLV-AR: A new iterative model that improves the knowledge review

Considering the information exposed in sections 5.2, 5.3 and 5.4, this one is devoted to describe a new iterative model combining pruned searching spaces and knowledge review in order to improve the interpretability of the rule base without loosing accuracy. Using this idea, the new iterative scheme is described below:

NEW_SEQUENTIAL_COVERING (E,F_δ,Δ)

1. Learned_rules ← {}
2. **For** i=1 to |Δ| **Do**
   (a) Action ← RETURN_ONE_ACTION (E, F_{δ_i}, Learned_rules, new_rule, rules_to_remove)
   (b) **While** (Action ≠ STOP) **Do**
      i. Learned_rules ← Add (Learned_rules, new_rule)
      ii. **If** (Action == REPLACE) **Then**

A. Learned_rules ← Remove (Learned_rules, rules_to_remove)

iii. E ← Penalize (Learned_rules, E)

iv. Action ← RETURN_ONE_ACTION (E, F_{δ_i}, Learned_rules, new_rule, rules_to_remove)

3. **Return** Learned_rules

with E the set of examples, F_δ the fitness function that depends on the δ parameter, Δ a decreasing sequence of δ values and RETURN_ONE_ACTION the procedure described in Section 5.2.1. The actions maintain the same behavior as the model previously described as well as the Add and Remove functions.

Finally, the fitness function supporting this proposal is:

$$fitness(R, \delta) = [\Psi_\delta(R),\ |\Omega(R)|,\ svar(R),\ sval(R)].$$

where $\Psi_\delta(R)$ is the product of consistency and completeness considering the parameter δ, $|\Omega(R)|$ is the number of replaceable rules, svar(R) is the number of irrelevant variables and sval(R) is the number of understandable assignments.

The algorithm acts searching for rules covering at least a percentage of examples of the class that is being learned, given by the parameter δ. The individuals are sorted according to their fitness function using a subpopulation for each class of the problem and an extra one for rules replacing rules. In each iteration three actions are possible (the algorithm decides which one is better among them): to add the best rule increasing the accuracy of the rule base, to add the best rule replacing rules or not to add any rule if the rule base can not be improved. The criterion for selecting the best option is maintained with regards to the one explained in Section 5.2.1.

5.6 Experimental study

Now, the information related to the experimental part is described. The algorithm aim of study is NSLV-AR, which has been deeply explained in previous sections. Following the same structure as those used in the other chapters, the results of the experimental study have been obtained using the configuration settings available in Chapter A. Due to the comparison between two classifiers, the statistical test used in this section have been the Wilcoxon's one. Some other important parameters related to the algorithms part of the analysis are detailed in Tables 5.1 and 5.2.

As was mentioned above, this algorithm focuses in the simplicity/interpretability criterion, so the parameters taken under consideration in the study are:

Table 5.1: Specific conditions for NSLV. NAV means the number of antecedent variables and *n_class* represents the number of classes of the specific problem.

	Specific Conditions NSLV
Size of genetic population	20*n_class
Number of iterations	500
Mutation prob. (Value level)	0.01
Mutation prob. (Variable level)	1/NAV
Mutation prob. (Consequent level)	0.01
Crossover prob.	1

Table 5.2: Specific conditions for NSLV-AR, where *n_class* represents the number of classes of the specific problem.

	Specific Conditions NSLV-AR
Size of genetic population	20*n_class
Number of iteration	500
δ_{max}	0.3
δ_{min}	0.1
Mutation prob. (Value level)	0.01
Mutation prob. (Consequent level)	0.01
Mutation prob. (Variable level)	1
Replacement prob.	0.05
Crossover prob.	0.95

- the accuracy on training,
- the accuracy on test,
- the average number of rules and
- the average number of conditions per rule base.

Thus, according to the Wilcoxon's test when using two classifiers and 40 datasets, and comparing the best algorithm in ranking (coincide with the best one in average) versus the other one (Table B.9), we observe that the null hypothesis is rejected in training (p-value $\leqslant \alpha$, Table 5.3), but not in test (Table 5.4). This means that NSLV obtains significantly the best performance in training with regard to NSLV-AR, but the results in test are quite similar. In fact, the average difference between training and test in NSLV-AR is around 5.8%, while in NSLV is near 8.8% ((Table B.9)).

Table 5.3: Results obtained by the Wilcoxon's test for algorithm NSLV (accuracy on training set), using $\alpha = 0.05$.

Vs	R^+	R^-	p-value	Hypothesis
NSLV-AR	741.0	79.0	0.000007	**Rejected**

Table 5.4: Results obtained by the Wilcoxon's test for algorithm NSLV (accuracy on testing set), using $\alpha = 0.05$.

Vs	R^+	R^-	p-value	Hypothesis
NSLV-AR	485.0	295.0	0.17575	**Not Rejected**

Table 5.5: Results obtained by the Wilcoxon's test for algorithm NSLV-AR (average number of rules), using $\alpha = 0.05$.

Vs	R^+	R^-	p-value	Hypothesis
NSLV	769.0	11.0	0	**Rejected**

Table 5.6: Results obtained by the Wilcoxon's test for algorithm NSLV-AR (average number of conditions per rule base), using $\alpha = 0.05$.

Vs	R^+	R^-	p-value	Hypothesis
NSLV	808.0	12.0	0	**Rejected**

If we center now in the results shown in tables 5.5, 5.6 and B.10, we can see that the null hypothesis is rejected, both in rules and conditions, which means that NSLV-AR significantly wins NSLV in both parameters. It is important to point out that, talking about average values, NSLV-AR reduces the number of rules almost half the number of rules achieved by NSLV. This fact also happens with regard to the number of conditions per rule base, where NSLV obtains more than twice the number of conditions achieved by NSLV-AR.

So, by making an overall view of the parameters included in the study, we realise that, despite significantly reducing the number of rules, NSLV-AR reaches similar values to NSLV in accuracy. This is important, because it means that NSLV-AR achieves a simpler knowledge without loosing prediction capability.

As occurred in the previous chapter, now we will briefly explain the differences among the rule bases obtained by each one of the proposals. Table 5.7 shows the accuracy on training and testing sets of both algorithms in the specific partition involved in the study.

Looking at figures 5.3 and 5.4 and the information given in Table 5.8, we can see that NSLV-AR gets a simpler rule base formed by a lower number of rules. It also obtains more interpretable rules, using less conditions. In this field, NSLV presents 44 conditions in its ruleset while NSLV-AR uses 32 conditions. Nevertheless, the last one achieves better prediction capability than NSLV obtaining 94.7% of accuracy on testing set against 92.9% of NSLV (Table 5.7).

Finally, Figure 5.2 gives a graphical representation of the behavior of

Table 5.7: Results obtained by the algorithms in the 9^{th} partition of the *wdbc* database. The table shows the accuracy on training and testing sets.

Algorithm	Training (%)	Test (%)
NSLV	98.6	92.9
NSLV-AR	98.4	94.7

Table 5.8: For each one of the algorithms, this table shows the number of successes/fails of each rule part of the different rule bases in the testing set.

	NSLV		NSLV-AR	
#Rule	Succ.	Fails	Succ.	Fails
R0	31	3	22	0
R1	12	1	1	1
R2	2	0	6	0
R3	0	0	0	0
R4	1	0	7	0
R5	3	0	6	1
R6	0	0	7	0
R7	3	0	5	1
R8	1	0	-	-

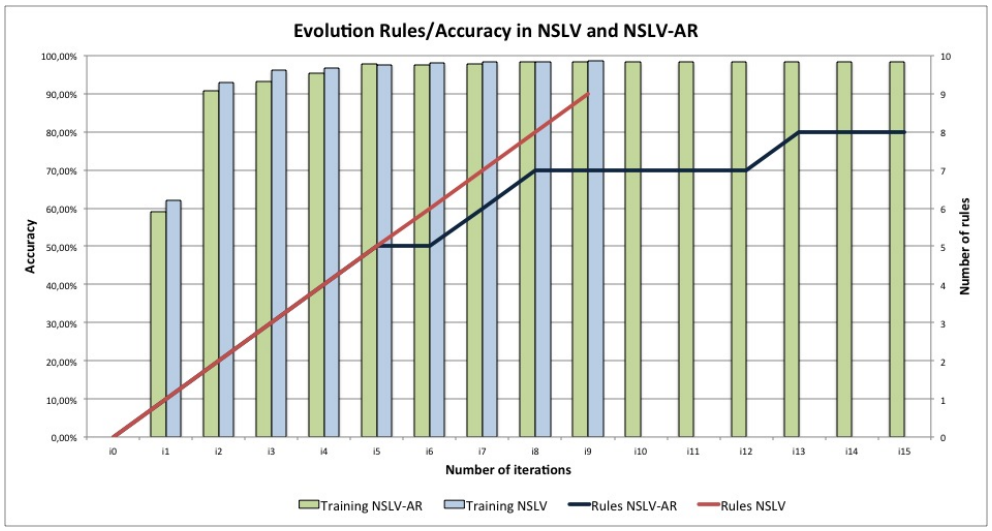


Figure 5.2: Evolution of accuracy and number of rules in each iteration of the learning process.

the algorithms during the learning process. It shows how the accuracy on training set and the number of rules evolve in each iteration.

5.7 Conclusions

This chapter presents a new iterative model able to review the learned knowledge while it is still learning. It also works on searching spaces pruned through a completeness condition in order to avoid extracting too specific rules which are almost useless to classify testing examples. Thus, this completeness condition is handled with a threshold (δ) moving on a decreasing sequence. This allows start searching more general rules in first stages of learning and moving to more specific ones in final stages. The threshold, in this situation, establishes the minimum percentage of examples that must be covered by a rule in order for it to be considered by the learning algorithm.

The experimental results demonstrate that the proposal (NSLV-AR) achieves simpler knowledge bases, improving the interpretability with regard to NSLV. It obtains significant differences in number of rules, reducing not only the number of rules per rule bases, but also the number of conditions per rule. It is also important to note that all these improvements are obtained without loosing accuracy. So, we finally have a model able to learn more interpretable rule bases with simpler rules and maintaining the prediction capability.

R0: IF Concave_points1 = { VeryLow Low} Area2 = { VeryLow} Area3 = { VeryLow} Compactness3 = { VeryLow Low} THEN Class IS B W 0.9279778959021685

R1: IF Perimeter3 = { Medium High VeryHigh} THEN Class IS M W 0.9427146601010222

R2: IF Smoothness1 = { Medium High VeryHigh} Compactness2 = { VeryLow Low High VeryHigh} Concavity2 = { VeryLow Low} Fractal_dimension2 = { VeryLow Low} Texture3 = { Medium High VeryHigh} Concave_points3 = { Medium High VeryHigh} THEN Class IS M W 0.8289296712126152

R3: IF Texture1 = { Medium VeryHigh} Symmetry1 = { Low} Compactness2 = { VeryLow Low High} Concavity2 = { VeryLow Low} Fractal_dimension2 = { VeryLow Low} Compactness3 = { Low} Concave_points3 = { Medium High VeryHigh} THEN Class IS M W 0.7284836582193538

R4: IF Texture1 = { VeryLow} Area1 = { VeryLow Low} Smoothness1 = { VeryLow Low VeryHigh} Perimeter2 = { VeryLow} Symmetry3 = { VeryLow Low} THEN Class IS B W 0.9877149519361487

R5: IF Texture1 = { Medium High VeryHigh} Concave_points2 = { Medium High VeryHigh} Area3 = { Low Medium VeryHigh} Concave_points3 = { VeryLow High VeryHigh} THEN Class IS M W 0.9624991605824593

R6: IF Radius1 = { Medium High} Texture1 = { High VeryHigh} Concave_points1 = { Low Medium High} Symmetry1 = { Medium High VeryHigh} Texture3 = { VeryLow Low High VeryHigh} THEN Class IS M W 0.9479497004775931

R7: IF Perimeter1 = { VeryLow Low} Compactness1 = { VeryLow Low} Symmetry1 = { VeryLow Low VeryHigh} Perimeter2 = { VeryLow High} Concave_points2 = { VeryLow Low} Texture3 = { VeryLow Low} Smoothness3 = { VeryLow Low VeryHigh} Concavity3 = { VeryLow Medium} Concave_points3 = { VeryLow Medium} Symmetry3 = { VeryLow Low} THEN Class IS B W 0.9761545147538883

R8: IF Concavity2 = { VeryLow Low} Symmetry3 = { High VeryHigh} THEN Class IS M W 1.0

Figure 5.3: Ruleset obtained by the algorithm NSLV in the 9^{th} partition of the wdbc database.

R0: IF Radius1 = { VeryLow Low} Area2 = { VeryLow} Concave_points3 = { VeryLow Low} THEN Class IS B W 0.9697728341533557
R1: IF Radius1 = { VeryLow Low High VeryHigh} Texture1 = { VeryLow Low} Radius2 = { VeryLow VeryHigh} Texture3 = { VeryLow Low} Area3 = { VeryLow} Symmetry3 = { Low} THEN Class IS B W 0.964425985101706
R2: IF Smoothness1 = { Medium High} Fractal_dimension2 = { VeryLow} Area3 = { Low Medium High VeryHigh} Smoothness3 = { Medium High VeryHigh} Concavity3 = { Medium High} THEN Class IS M W 0.9573650565508308
R3: IF Texture1 = { VeryLow} Area1 = { Low} Compactness1 = { Low High VeryHigh} Concavity1 = { VeryLow Low} Perimeter2 = { VeryLow} THEN Class IS B W 0.9777279010275604
R4: IF Fractal_dimension1 = { VeryLow Low High VeryHigh} Radius3 = { VeryLow} Symmetry3 = { Low Medium} THEN Class IS B W 0.9918865510091466
R5: IF Radius1 = { Medium} Concave_points3 = { VeryLow Low High VeryHigh} THEN Class IS M W 0.7206203310773723
R6: IF Perimeter1 = { Low Medium VeryHigh} Area3 = { Low Medium High VeryHigh} Smoothness3 = { Medium VeryHigh} Concavity3 = { Low Medium} THEN Class IS M W 0.5909202519518316
R7: IF Concavity1 = { VeryLow} Fractal_dimension1 = { Low Medium High VeryHigh} Texture2 = { VeryLow} Texture3 = { VeryLow Low} THEN Class IS B W 0.9709823485535519

Figure 5.4: Ruleset obtained by the algorithm NSLV-AR in the 9^{th} partition of the wdbc database.

Chapter 6

An integrated method using feature construction and knowledge review

6.1 Introduction and foundations

According to the structure of this dissertation, two main branches can be distinguished: on the one hand the proposals related to feature construction, mainly focused on the improvement of the prediction capability of the models. On the other hand, the proposal related to knowledge review, more focused on interpretability. Although both ideas are contrary related (normally, the improvement of one of them implies the decrease of the other), now the objective is to integrate them in an algorithm in order to find a good trade-off among accuracy and interpretability.

The feature construction techniques normally increase the searching space of possible solutions. The use of relations and functions give a tool for handling additional information apart from the one given by the original variables. As a consequence, the prediction capability of the model obtained using this approach, is normally increased, because some more examples can be represented by the set of rules. The problem is that some of these examples are isolated or very specific and do not give useful information for classifying test examples.

The knowledge review can be seen as a refinement process which specially in final stages of learning, helps replacing those rules which can be

improved by a later one. This mechanism improves the rule base from an interpretability point of view without loosing accuracy. The combination of this idea together with the use of pruned searching spaces allows defining the specificity level of the learned rules. So, both of them, the interpretability at a rule base level and for each single rule, are improved.

Thus, the proposal described in this chapter, called SLAVE3 in order to recover the line previously established by SLAVE and SLAVE2, aims to be an algorithm that integrates the techniques aforementioned. The main objective is to achieve a good trade-off among accuracy and interpretability. In this sense, it includes feature construction through the use of relations and functions and also knowledge review together with pruned searching spaces. Thereby, it maintains the same ideas as the rest of proposals following the IRL approach and applying all these techniques embedded in the learning process.

A global view of the steps given until reaching this version, SLAVE3, is shown in Figure 6.1.

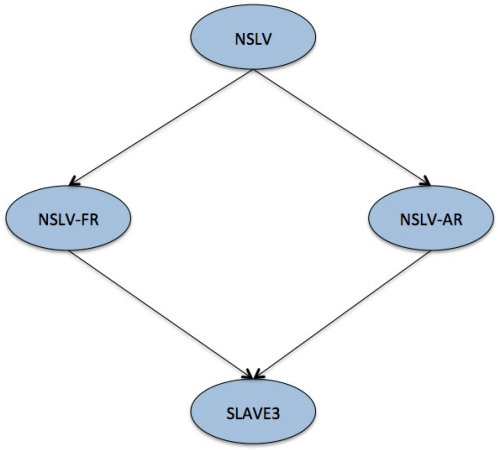


Figure 6.1: Steps given until reaching SLAVE3.

In this way, SLAVE3 uses the same rule codification as the one described in Section 4.4, Figure 4.10, namely, that employed to take into account both relations and functions in the antecedent of fuzzy rules. It also inherits the same rule model and the same genetic operators as those used by NSLV-FR, together with the replacement operator introduced when talked about knowledge review (Section 5.2.1).

With regard to the fitness function, the proposed method considers the one given below:

$$fitness(R) = [\Psi_\delta(R),\ |\Omega(R)|,\ svar(R),\ sval(R)]$$

being:

- $\Psi_\delta(R)$: the first component of the fitness function, expressed as:

$$\Psi_\delta(R) = \begin{cases} (\Gamma'_{k_1,k_2}(R) \times \Lambda(R) \times w^n) & \text{if } (\Lambda(R) \geq \delta) \\ -\infty & \text{otherwise.} \end{cases} \quad (6.1)$$

- $\Gamma'_{k_1,k_2}(R)$: the consistency definition given in Section 4.4.1.
- $\Lambda(R)$: the completeness degree definition detailed in Section 3.1.3.2.2.
- w: the weight of the rule.
- n: the number of conditions, understanding as conditions the initial variables, relations and functions considered in the antecedent of a rule.
- $svar(R)$: the simplicity degree of a rule determined by the number of irrelevant variables. The expression is detailed in Section 3.2.2.3.3, equation 3.31.
- $sval(R)$: the simplicity degree of a rule determined by the number of understandable assignments to the variables of a rule. The expression is shown in Section 3.2.2.3.3, equation 3.32.
- δ: the threshold that represents the minimum percentage of examples (of the class that is being learned), that must be covered by a rule.

Finally, one important characteristic of SLAVE3 is its ability to be configured in order to behave as the methods that integrate it. In this sense, we are able to decide whether to consider relations, functions or the reviewing functionality.

Next section is devoted to describe a wide experimental study including the main proposals presented in this work.

6.2 Experimental study

6.2.1 Comparison among all proposals

In this subsection, SLAVE3 is compared with the rest of proposals described in the dissertation. The objective of this comparison is to demonstrate that the behavior of each model is consistent with the storyline. On the

other hand, the idea here is to show the SLAVE3 learning algorithm as the final model including the main advantages of the previous algorithms but minimizing their weaknesses. With this purpose, the statistical test that allows this comparative study is the Shaffer's test.

One more time, the results in this experimental analysis have been obtained considering the general settings mentioned in Chapter A. Some other specific conditions associated to the different algorithms are shown in tables 6.1, 6.2 and 6.3.

Table 6.1: Specific conditions for NSLV and NSLV-FR. NAV means the number of antecedent variables and *n_class* represents the number of classes of the specific problem.

	Specific Conditions NSLV/NSLV-FR
Size of genetic population	20*n_class
Number of iterations	500
Mutation prob. (Value level)	0.01
Mutation prob. (Variable level)	1/NAV
Mutation prob. (Consequent level)	0.01
Crossover prob.	1

Table 6.2: Specific conditions for NSLV-AR, where *n_class* represents the number of classes of the specific problem.

	Specific Conditions NSLV-AR
Size of genetic population	20*n_class
Number of iteration	500
δ_{max}	0.3
δ_{min}	0.1
Mutation prob. (Value level)	0.01
Mutation prob. (Consequent level)	0.01
Mutation prob. (Variable level)	1
Replacement prob.	0.05
Crossover prob.	0.95

The parameters in which the experimental study is based are listed below:

- the accuracy on training set,
- the accuracy on testing set,
- the average number of rules and
- the average number of conditions per rule.

Table 6.3: Specific conditions for SLAVE3, where *n_class* represents the number of classes of the specific problem.

	Specific Conditions SLAVE3
Size of genetic population	20*n_class
Number of iteration	500
δ_{max}	0.3
δ_{min}	0.1
Mutation prob. (Value level)	0.01
Mutation prob. (Consequent level)	0.01
Mutation prob. (Variable level)	1
Replacement prob.	0.05
Crossover prob.	0.95

According to all these considerations and as it is usual in all the experimental analysis made up to now, we will describe the results using the tables given by the statistical tests.

Table 6.4: Average Rankings of the algorithms (Friedman) and computed p-value by Friedman (accuracy on training set).

Algorithm	Ranking
NSLV	2.6375
NSLVAR	3.675
NSLVFR	1.3375
SLAVE3	2.35

Friedman p-value
4.429012712137137E-11

Table 6.5: Adjusted p-values (accuracy on training set).

i	hypothesis	unadjusted p	p_{Shaf}
1	NSLVAR vs .NSLVFR	0	0
2	NSLVAR vs .SLAVE3	0.000004	0.000013
3	NSLV vs .NSLVFR	0.000007	0.00002
4	NSLV vs .NSLVAR	0.000326	0.000977
5	NSLVFR vs .SLAVE3	0.000453	0.000977
6	NSLV vs .SLAVE3	0.319285	0.319285

So, looking at Table 6.5, we can see that all the algorithms obtain significant differences in training with regard to NSLV-AR (the worst algorithm in ranking), and the same occurs among NSLV-FR and the rest of proposals. On the other hand, in relation to Table 6.7, we observe that all the methods including feature construction (NSLV-FR and SLAVE3), significantly win NSLV-AR in test (and both are close to obtain statistical differences also with NSLV, but considering a significance level of $\alpha = 0.058$ and $\alpha = 0.113$, respectively). Regarding to ranking classification and average values (tables 6.4, 6.6, B.11 and B.12), it is proved that the methods using the feature

construction techniques present the best prediction capability with similar results in test. In this situation, SLAVE3 is the second best algorithm in accuracy, very near to NSLV-FR which is the one achieving the best results.

Table 6.6: Average Rankings of the algorithms (Friedman) and computed p-values by Friedman and Iman-Davenport (accuracy on testing set).

Algorithm	Ranking
NSLV	2.7625
NSLVAR	2.9875
NSLVFR	2.0875
SLAVE3	2.1625

Friedman p-value
0.0026763709780885936

Table 6.7: Adjusted p-values (accuracy on testing set).

i	hypothesis	unadjusted p	p_{Shaf}
1	NSLVAR vs .NSLVFR	0.001823	0.010936
2	NSLVAR vs .SLAVE3	0.004265	0.012794
3	NSLV vs .NSLVFR	0.019373	0.05812
4	NSLV vs .SLAVE3	0.037667	0.113001
5	NSLV vs .NSLVAR	0.435731	0.871461
6	NSLVFR vs .SLAVE3	0.795012	0.871461

Table 6.8: Average Rankings of the algorithms (Friedman) and computed p-values by Friedman and Iman-Davenport (average number of rules).

Algorithm	Ranking
NSLV	2.75
NSLVAR	1.3625
NSLVFR	3.225
SLAVE3	2.6625

Friedman p-value
6.764379056889425E-10

Now, focusing in Table 6.10, we observe that NSLV-AR significantly wins the rest of algorithms in number of rules, something expected as it is the only one including the knowledge review technique alone. It is the best one also when talking about average values and ranking classification (tables 6.8 and B.13). It is important to note that SLAVE3, which also includes knowledge review, is the second best algorithm, although it obtains almost twice the average number of rules achieved by NSLV-AR. Anyway, it is comprehensible because SLAVE3 also uses feature construction which is susceptible to increase the knowledge base.

Table 6.9: Average Rankings of the algorithms (Friedman) and computed p-values by Friedman and Iman-Davenport (average number of conditions per rule).

Algorithm	Ranking
NSLV	3.2875
NSLVAR	1.65
NSLVFR	2.875
SLAVE3	2.1875

Friedman p-value
2.9091349995624682E-8

The same happens with the last parameter, shown in Table 6.11. The number of conditions per rule is significantly reduced by the proposals including knowledge review (NSLV-AR and SLAVE3). In this sense, NSLV-AR is again the best algorithm, closely followed by SLAVE3 (in average values, Table B.14). We can also appreciate that SLAVE3 is near to obtain statistical differences with NSLV-FR, something that would be possible with a significance level of 0.052.

Table 6.10: Adjusted p-values (average number of rules).

i	hypothesis	unadjusted p	p_{Shaf}
1	NSLVAR vs .NSLVFR	0	0
2	NSLV vs .NSLVAR	0.000002	0.000005
3	NSLVAR vs .SLAVE3	0.000007	0.00002
4	NSLVFR vs .SLAVE3	0.051348	0.154045
5	NSLV vs .NSLVFR	0.099877	0.199755
6	NSLV vs .SLAVE3	0.761807	0.761807

Table 6.11: Adjusted p-values (average number of conditions per rule).

i	hypothesis	unadjusted p	p_{Shaf}
1	NSLV vs .NSLVAR	0	0
2	NSLVAR vs .NSLVFR	0.000022	0.000066
3	NSLV vs .SLAVE3	0.000139	0.000416
4	NSLVFR vs .SLAVE3	0.017239	0.051717
5	NSLVAR vs .SLAVE3	0.062609	0.125219
6	NSLV vs .NSLVFR	0.153021	0.153021

Therefore, globally, SLAVE3 successfully achieves a good trade-off among accuracy and interpretability. In fact, SLAVE3 proves to be a learning algorithm that successfully combines feature construction techniques with knowledge review, minimizing their main disadvantages.

In order to show the different rule bases obtained by each algorithm, next we will present some examples for the *wdbc* database (Figure A.1). Table 6.12 shows the accuracy of the different proposals when running the 9^{th} partition of the *wdbc* dataset.

Paying attention to figures 6.2, 6.3, 6.4 and 6.5 and Table 6.12, we can see

Table 6.12: Results obtained by the algorithms in the 9^{th} partition of the *wdbc* database. The table shows the accuracy on training and testing sets.

Algorithm	Training (%)	Test (%)
NSLV	98.6	92.9
NSLV-AR	98.4	94.7
NSLV-FR	98.8	98.2
SLAVE3	97.4	96.4

that the conclusions extracted in the previous analysis are maintained. Thus, the higher accuracy is again achieved by NSLV-FR, followed by SLAVE3. However, contrary to the results shown in Table B.13, in this particular set of examples NSLV-FR obtains the rule base with lower number of rules. Anyway, the simpler rule base is just one measure of interpretability. When looking at Table 6.14, we realize that the algorithm with simpler rules is SLAVE3. So, one more time, SLAVE3 demonstrate having a good trade-off accuracy/interpretability.

Table 6.13: For each one of the algorithms, this table shows the number of successes/fails of each rule part of the different rule bases in the testing set.

	NSLV		NSLV-AR		NSLV-FR		SLAVE3	
#Rule	Succ.	Fails	Succ.	Fails	Succ.	Fails	Succ.	Fails
R0	31	3	22	0	30	0	32	2
R1	12	1	1	1	16	0	6	0
R2	2	0	6	0	3	0	1	0
R3	0	0	0	0	0	1	4	1
R4	1	0	7	0	5	0	1	0
R5	3	0	6	1	2	0	8	0
R6	0	0	7	0	-	-	3	0
R7	3	0	5	1	-	-	-	-
R8	1	0	-	-	-	-	-	-

Table 6.14: Results obtained by the algorithms in the 9^{th} partition of the *wdbc* database. The table shows the average number of conditions per rule.

Algorithm	Conditions per rule
NSLV	4.88
NSLV-AR	3
NSLV-FR	4
SLAVE3	2.71

Regarding to the results appearing in Table 6.13, it is important to say that, as occurs with the rest of algorithms, the first rule of SLAVE3 is the one with higher successes index. The second one according to the number of successes is $R5$, which uses two fuzzy relations and obtains the higher

weight of the whole ruleset.

R0: IF Concave_points1 = { VeryLow Low} Area2 = { VeryLow} Area3 = { VeryLow} Compactness3 = { VeryLow Low} THEN Class IS B W 0.9279778959021685
R1: IF Perimeter3 = { Medium High VeryHigh} THEN Class IS M W 0.9427146601010222
R2: IF Smoothness1 = { Medium High VeryHigh} Compactness2 = { VeryLow Low High VeryHigh} Concavity2 = { VeryLow Low} Fractal_dimension2 = { VeryLow Low} Texture3 = { Medium High VeryHigh} Concave_points3 = { Medium High VeryHigh} THEN Class IS M W 0.8289296712126152
R3: IF Texture1 = { Medium VeryHigh} Symmetry1 = { Low} Compactness2 = { VeryLow Low High} Concavity2 = { VeryLow Low} Fractal_dimension2 = { VeryLow Low} Compactness3 = { Low} Concave_points3 = { Medium High VeryHigh} THEN Class IS M W 0.7284836582193538
R4: IF Texture1 = { VeryLow} Area1 = { VeryLow Low} Smoothness1 = { VeryLow Low VeryHigh} Perimeter2 = { VeryLow} Symmetry3 = { VeryLow Low} THEN Class IS B W 0.9877149519361487
R5: IF Texture1 = { Medium High VeryHigh} Concave_points2 = { Medium High VeryHigh} Area3 = { Low Medium VeryHigh} Concave_points3 = { VeryLow High VeryHigh} THEN Class IS M W 0.9624991605824593
R6: IF Radius1 = { Medium High} Texture1 = { High VeryHigh} Concave_points1 = { Low Medium High} Symmetry1 = { Medium High VeryHigh} Texture3 = { VeryLow Low High VeryHigh} THEN Class IS M W 0.9479497004775931
R7: IF Perimeter1 = { VeryLow Low} Compactness1 = { VeryLow Low} Symmetry1 = { VeryLow Low VeryHigh} Perimeter2 = { VeryLow High} Concave_points2 = { VeryLow Low} Texture3 = { VeryLow Low} Smoothness3 = { VeryLow Low VeryHigh} Concavity3 = { VeryLow Medium} Concave_points3 = { VeryLow Medium} Symmetry3 = { VeryLow Low} THEN Class IS B W 0.9761545147538883
R8: IF Concavity2 = { VeryLow Low} Symmetry3 = { High VeryHigh} THEN Class IS M W 1.0

Figure 6.2: Ruleset obtained by the algorithm NSLV in the 9^{th} partition of the wdbc database.

R0: IF Radius1 = { VeryLow Low} Area2 = { VeryLow} Concave_points3 = { VeryLow Low} THEN Class IS B W 0.9697728341533557
R1: IF Radius1 = { VeryLow Low High VeryHigh} Texture1 = { VeryLow Low} Radius2 = { VeryLow VeryHigh} Texture3 = { VeryLow Low} Area3 = { VeryLow} Symmetry3 = { Low} THEN Class IS B W 0.964425985101706
R2: IF Smoothness1 = { Medium High} Fractal_dimension2 = { VeryLow} Area3 = { Low Medium High VeryHigh} Smoothness3 = { Medium High VeryHigh} Concavity3 = { Medium High} THEN Class IS M W 0.9573650565508308
R3: IF Texture1 = { VeryLow} Area1 = { Low} Compactness1 = { Low High VeryHigh} Concavity1 = { VeryLow Low} Perimeter2 = { VeryLow} THEN Class IS B W 0.9777279010275604
R4: IF Fractal_dimension1 = { VeryLow Low High VeryHigh} Radius3 = { VeryLow} Symmetry3 = { Low Medium} THEN Class IS B W 0.9918865510091466
R5: IF Radius1 = { Medium} Concave_points3 = { VeryLow Low High VeryHigh} THEN Class IS M W 0.7206203310773723
R6: IF Perimeter1 = { Low Medium VeryHigh} Area3 = { Low Medium High VeryHigh} Smoothness3 = { Medium VeryHigh} Concavity3 = { Low Medium} THEN Class IS M W 0.5909202519518316
R7: IF Concavity1 = { VeryLow} Fractal_dimension1 = { Low Medium High VeryHigh} Texture2 = { VeryLow} Texture3 = { VeryLow Low} THEN Class IS B W 0.9709823485535519

Figure 6.3: Ruleset obtained by the algorithm NSLV-AR in the 9^{th} partition of the wdbc database.

R0: IF Area3 = { VeryLow} Concave_points3 = { VeryLow Low} THEN Class IS B W 0.9749258286226512
R1: IF (Radius1/Area3) = { L0 L2} Fractal_dimension1 < Concave_points1 Smoothness1 < Concavity1 THEN Class IS M W 0.9998451137419374
R2: IF Texture3 = { Medium High VeryHigh} (Radius1/Area3) = { L0} Concave_points3 ! =E Compactness2 Fractal_dimension1 ⩽E Concavity1 THEN Class IS M W 0.9732671722874883
R3: IF Compactness2 = { VeryLow Medium High VeryHigh} Radius3 = { Medium VeryHigh} Texture3 = { Medium High VeryHigh} THEN Class IS M W 0.9490180336074999
R4: IF Symmetry1 = { VeryLow Medium VeryHigh} Texture3 = { VeryLow Low} (Compactness1-Perimeter3) = { L1 L2 L3} THEN Class IS B W 0.8258497508137139
R5: IF (Concavity2-Radius3) = { L1 L2 L3 L4} (Radius3-Perimeter3) = { L1 L2 L3 L4} Concave_points2 ! =E Concave_points1 THEN Class IS M W 0.9387358486204522

Figure 6.4: Ruleset obtained by the algorithm NSLV-FR in the 9^{th} partition of the wdbc database.

R0: IF Concave_points3 = { MuyBaja Baja} THEN Class IS B W 0.9450622665520259
R1: IF Concave_points3 = { Alta MuyAlta} Concave_points1 ! =E Concave_points2 THEN Class IS M W 0.9858711171095856
R2: IF Texture3 = { Media MuyAlta} (Smoothness2+Perimeter3) = { L2 L3} THEN Class IS M W 0.9715859133303734
R3: IF Texture3 = { Media MuyAlta} Fractal_dimension1 < Concavity1 Area2 ! =E Radius1 Smoothness1 < Concave_points3 THEN Class IS M W 0.9762838732075463
R4: IF Texture1 = { MuyBaja Baja Alta} Texture2 = { MuyBaja} (Concave_points1*Concavity3) = { L2 L3} Concavity1 < Compactness1 Area1 =E Area3 THEN Class IS B W 0.9907741137371731
R5: IF Fractal_dimension2 = { MuyBaja} Texture3 = { Media} Concavity1 ! =E Compactness2 Concave_points1 ! =E Concave_points2 THEN Class IS M W 0.9955177092569343
R6: IF Concave_points1 ⩽E Concavity2 THEN Class IS B W 0.936713110626154

Figure 6.5: Ruleset obtained by the algorithm SLAVE3 in the 9^{th} partition of the wdbc database.

6.2.2 SLAVE3 Vs other learning algorithms

This section is devoted to compare the last method proposed in this work, that is, SLAVE3, against some well-known classical algorithms and some more recent proposals. Some of the algorithms involved in the comparison have been properly described in Chapter 2. The rest of them are briefly summarized below:

- C4.5([95]): Proposed by R.S. Quinlan, it is an inductive classification algorithm that represents knowledge using a decision tree.
- FURIA ([60]): It is a fuzzy rule-based classifier proposed by Hühn and Hüllermeier, which is an advancement of the RIPPER ([19]) algorithm, differing from the last one mainly in the use of fuzzy instead of conventional rules. It also introduces a rule stretching technique to reduce the computational complexity improving the performance.
- FARC-HD ([2]): Proposed by J. Alcalá et al., it is a fuzzy associative classification method for high-dimensional datasets. It is based on three stages to obtain accurate and compact fuzzy rule bases: the first one is related to fuzzy association rule extraction for classification, the second one is devoted to preselect the most interesting rules from the rule base obtained in the previous stage and finally, the third one is responsible of rule selection and lateral tuning.

This time, we have used the Shaffer's statistical test to be able to compare all possible pairwise. All methods have been run through the KEEL platform and using the default settings. SLAVE3 maintains the same specific conditions as shown in Table 6.3. The rest of considerations related to the experimental settings can be consulted in Chapter A.

Table 6.15: Average Rankings of the algorithms (Friedman) and computed p-value by Friedman (accuracy on training set).

Algorithm	Ranking
GCCL	5.225
C45	2.4625
SGERD	5.55
FARC-HD	2.125
FURIA	2.9
SLAVE3	2.7375

Friedman p-value
5.778677536483201E-11

Now, the study focuses in four parameters: the accuracy on training and testing sets, the average number of rules and the average trade-off measured as the ratio given by the accuracy on test and the number of rules. In this section, the study has not been extended to the number of conditions,

as some of the algorithms did not give that parameter available. Anyway, this situation does not constitute a penalty and the purpose of the study is successfully achieved.

First of all and starting with the accuracy on training set we can see in Table 6.15 that the best algorithm in ranking is FARC-HD, closely followed by SLAVE3. According to the Friedman p-value, some statistical differences are found. In this sense, Table 6.16 shows that FARC-HD, SLAVE3, FURIA and C4.5 achieve significant differences when compared to GCCL and SGERD, but not among them. In average values (Table B.15), C4.5 presents the best results with FARC-HD and SLAVE3 moving in similar ranges.

Table 6.16: Adjusted p-values (accuracy on training set).

i	hypothesis	unadjusted p	p_{Shaf}
1	SGERD vs .FARC-HD	0	0
2	GCCL vs .FARC-HD	0	0
3	C45 vs .SGERD	0	0
4	SGERD vs .SLAVE3	0	0
5	GCCL vs .C45	0	0
6	SGERD vs .FURIA	0	0
7	GCCL vs .SLAVE3	0	0
8	GCCL vs .FURIA	0	0
9	FARC-HD vs .FURIA	0.063939	0.447574
10	FARC-HD vs .SLAVE3	0.143152	0.85891
11	C45 vs .FURIA	0.295642	1.182567
12	C45 vs .FARC-HD	0.419794	1.679175
13	GCCL vs .SGERD	0.437219	1.679175
14	C45 vs .SLAVE3	0.510939	1.679175
15	FURIA vs .SLAVE3	0.697684	1.679175

Table 6.17: Average Rankings of the algorithms (Friedman) and computed p-values by Friedman and Iman-Davenport (accuracy on testing set).

Algorithm	Ranking
GCCL	4.8
C45	3.05
SGERD	5.2125
FARC-HD	2.675
FURIA	2.25
SLAVE3	3.0125

Friedman p-value
4.43937109295689E-11

Regarding to the test parameter, in Table 6.17 we observe that the ranking classification changes in relation to the training parameter. Now, the best method is FURIA and SLAVE3 holds the third position. Again, the Friedman p-value indicates statistical differences among the algorithms. As occurred before, FURIA, FARC-HD, SLAVE3 and C4.5 obtain statistical

differences with GCCL and SGERD (Table 6.18). No significant differences are found among them. If looking at average values (Table B.16), the best algorithm keeps being FURIA but SLAVE3 becomes the second best one (slightly better than FARC-HD).

Table 6.18: Adjusted p-values (accuracy on testing set).

i	hypothesis	unadjusted p	p_{Shaf}
1	SGERD vs .FURIA	0	0
2	GCCL vs .FURIA	0	0
3	SGERD vs .FARC-HD	0	0
4	SGERD vs .SLAVE3	0	0.000001
5	C45 vs .SGERD	0	0.000002
6	GCCL vs .FARC-HD	0	0.000004
7	GCCL vs .SLAVE3	0.000019	0.000135
8	GCCL vs .C45	0.000029	0.000201
9	C45 vs .FURIA	0.055829	0.390805
10	FURIA vs .SLAVE3	0.068345	0.410072
11	FARC-HD vs .FURIA	0.309656	1.238624
12	GCCL vs .SGERD	0.324102	1.296408
13	C45 vs .FARC-HD	0.370028	1.296408
14	FARC-HD vs .SLAVE3	0.419794	1.296408
15	C45 vs .SLAVE3	0.928572	1.296408

Table 6.19: Average Rankings of the algorithms (Friedman) and computed p-values by Friedman and Iman-Davenport (average number of rules).

Algorithm	Ranking
GCCL	4.8
C45	3.5
SGERD	1.2
FARC-HD	4.8
FURIA	3.4
SLAVE3	3.3

Friedman p-value
6.764379056889425E-10

Looking at Table 6.19, we realize that the algorithm achieving fewer number of rules is SGERD. This algorithm obtains statistical differences with the rest of methods involved in the comparison (Table 6.20). SLAVE3, FURIA and C4.5 present similar results, not only in ranking, but also in average values (Table B.17). In this sense, they significantly win FARC-HD and GCCL, but without differences among them. Considering the average results, GCCL (the worst algorithm), obtains almost nine times more rules than SGERD (the best one). The rest (but FARC-HD), get similar values in average, reducing the number of rules almost in 20 rules per dataset (when compared with GCCL).

Table 6.20: Adjusted p-values (average number of rules).

i	hypothesis	unadjusted p	p_{Shaf}
1	GCCL vs .SGERD	0	0
2	SGERD vs .FARC-HD	0	0
3	C45 vs .SGERD	0	0
4	SGERD vs .FURIA	0	0.000001
5	SGERD vs .SLAVE3	0.000001	0.000005
6	GCCL vs .SLAVE3	0.000336	0.003362
7	FARC-HD vs .SLAVE3	0.000336	0.003362
8	GCCL vs .FURIA	0.000818	0.005726
9	FARC-HD vs .FURIA	0.000818	0.005726
10	GCCL vs .C45	0.001886	0.011317
11	C45 vs .FARC-HD	0.001886	0.011317
12	C45 vs .SLAVE3	0.632585	2.53034
13	FURIA vs .SLAVE3	0.81107	2.53034
14	C45 vs .FURIA	0.81107	2.53034
15	GCCL vs .FARC-HD	1	2.53034

Finally, considering as a new parameter the trade-off between accuracy (in test) and interpretability (measured through the number of rules), we can make the following study. First, we have to point out that this trade-off have been measured following the criterion $accuracyTest/numberRules$ for each single database.

Table 6.21: Average Rankings of the algorithms (Friedman) and computed p-values by Friedman and Iman-Davenport (average trade-off accuracy (test)/interpretability (rules)).

Algorithm	Ranking
GCCL	5.2
C45	3.375
SGERD	1.25
FARC-HD	4.725
FURIA	3.3
SLAVE3	3.15

Friedman p-value
6.529232710050792E-11

So, according to the results shown in Table 6.21, we appreciate that the algorithm obtaining the best trade-off accuracy/interpretability is SGERD. The next one is SLAVE3, closely followed by FURIA and C4.5. With more details, we can see in Table 6.22, that the same conclusions as those given for the rules parameter, are maintained. So, again, SGERD is significantly better than the rest of proposals.

Thus, in summary, we can say that SLAVE3 achieves good results, belonging to the group of the algorithms with higher prediction capability, with no significant differences with respect to the best ones (FARC-HD or FURIA,

Table 6.22: Adjusted p-values (average trade-off accuracy (test)/interpretability (rules)).

i	hypothesis	unadjusted p	p_{Shaf}
1	GCCL vs .SGERD	0	0
2	SGERD vs .FARC-HD	0	0
3	C45 vs .SGERD	0	0.000004
4	SGERD vs .FURIA	0.000001	0.00001
5	GCCL vs .SLAVE3	0.000001	0.00001
6	SGERD vs .SLAVE3	0.000006	0.000056
7	GCCL vs .FURIA	0.000006	0.000056
8	GCCL vs .C45	0.000013	0.00009
9	FARC-HD vs .SLAVE3	0.000167	0.001166
10	FARC-HD vs .FURIA	0.000658	0.00395
11	C45 vs .FARC-HD	0.00125	0.005002
12	GCCL vs .FARC-HD	0.25618	1.024719
13	C45 vs .SLAVE3	0.590679	1.772036
14	FURIA vs .SLAVE3	0.719918	1.772036
15	C45 vs .FURIA	0.857714	1.772036

depending on the training/test parameter). It also presents good classifications in number of rules and trade-off accuracy/interpretability, being the second best algorithm in both parameters. In these cases, SGERD obtains the best results and the rest of methods are far away from it. Anyway, we have to take into account that SGERD is also one of the worst algorithms in prediction capability, being a 10% worse in average than SLAVE3. This is important, because the simplicity in number of rules of the rule bases generated by SGERD, translates in an important loss of accuracy. As a consequence, we can conclude that SLAVE3 has a good general behavior, also when dealing with complex problems.

6.2.3 A first step towards the incremental learning

6.2.3.1 Motivation and general purpose

During the development of this dissertation, the main efforts have focused in obtaining fuzzy rule-based systems describing the examples of a particular problem. Normally, these problems have been represented through databases formed by a set of pre-selected examples which do not change along time. Therefore, it can be considered that all methods exposed up to now exclusively make an offline learning. This means that each time a fuzzy system representing changing examples is required, the whole learning process should be repeated.

Some of the real problems today move in a changing environment. This implies continuously considering new information and under a learning point of view, to re-adapt the knowledge. This situation is commonly associated

in the bibliography to the terms: incremental learning, online learning or concept drift.

The purpose in this section is to give a first step towards this capability of re-adapting the learned knowledge. In this sense, the main purpose is to check the capability of SLAVE3 to behave on an incremental way. Following this idea, we will assume that the algorithm stores all the training cases and it is able to handle the examples used in the previous learning stage.

Therefore, we will employ the SLAVE3 learning algorithm but disabling the use of relations and functions in the antecedent of the rules (because the use of relations and functions increase both the learning complexity and time requirements, and because we are also focused on interpretability measures more related to the knowledge review), and we will refer to it as SLAVE3*. Then, the idea is to work in several steps considering increments of information given by the training examples. Two main tasks are now taken into account. The first one is related to the way the ruleset is managed, as in each new step the previously learned ruleset is considered as the new initial ruleset. The second change is related to the update of the weights of each initial ruleset but using the new set of examples.

Now, let us imagine an initial situation in which the algorithm handles a training set $E_0 = E$ and a ruleset $R_0 = \emptyset$ and obtains a set of rules R_1. So, if at any time the number of examples increases, $E_1 = E_0 + \tau_1$, being τ_1 an incremental set of examples, then the algorithm would learn by using the training set E_1 and the input rule base R_1. Here, the sub-indexes should be understood as an ordered sequence of time steps (i.e., $R_0 = R_t$, with $t = 0$). Thereby, the process can then continue while new increments of examples arrive.

With regard to the selection of the initial population each time a new training set is added, it is important to point out that the initial population in the time step $t + 1$ is obtained by randomly taking the misclassified examples by the current rule base R_t with the new set of examples E_{t+1}.

6.2.3.2 Experimental study

Up to now, we have been working with offline data, and we still do, so with the purpose of checking the behavior of the proposal, two different experiments have been conducted. In the first one, we analyze the evolution of the accuracy in some particular databases when the training set is increased using short increments. In the second experimental part, we consider only two steps, the first one using the 90% of the training data, and the second one adding the remaining 10%. This second configuration is then compared with the original (non-incremental) version using the whole list of databases described in Table A.1. Figure 6.6 shows a graphic scheme of the behavior of the proposed model in this experimental part.

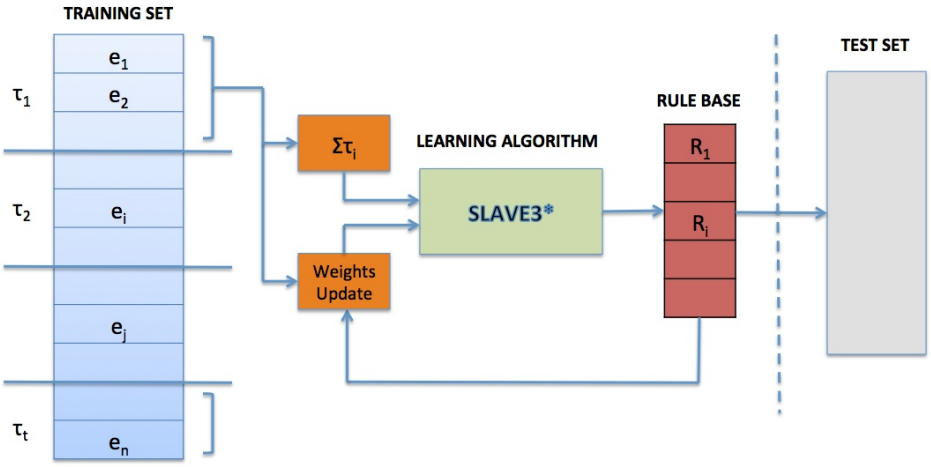


Figure 6.6: Example of how SLAVE3* acts in this experimental study.

Another important point regarding to the definition of the incremental example sets used in this experimentation is that they must satisfy the following conditions:

- $E_0 \cap E_1 \cap ... \cap E_n = \emptyset$
- $E_0 \cup E_1 \cup ... \cup E_n = E$.

6.2.3.2.1 First experiment

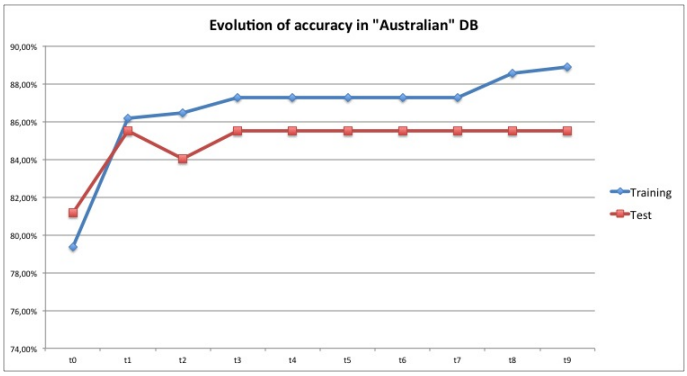
In this first subsection, we analyze how the accuracy evolves in three particular datasets using short increments of the training sets. These increments act as correlated time sequences.

The selected datasets in the study have been **australian**, **dermatology** and **wisconsin**, which are described in Table A.1. The training sets of each one of these databases have been divided into ten disjunct subsets with approximately the same number of examples (10% of the total set of examples).

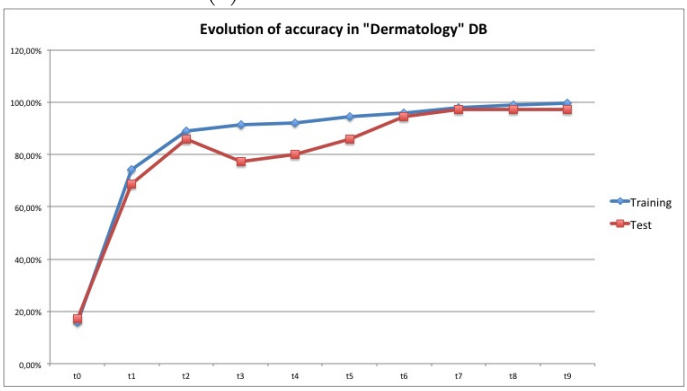
The evolution followed by the algorithm in each database (learning curve) is graphically shown in Figures 6.7(a), 6.7(b) and 6.7(c), where t_i represents the moment in which the incremental set of examples E_i is considered.

According to these representations, we can conclude that the behavior in the three databases is not exactly the same, but there are some similar components. Specifically, there is a very quick growth of the accuracy in all cases, later on it is gradually improved and finally it makes stable. This behavior seems desirable in a process of incremental learning. This "stable" point can be considered as the point in which the learned model is reliable.

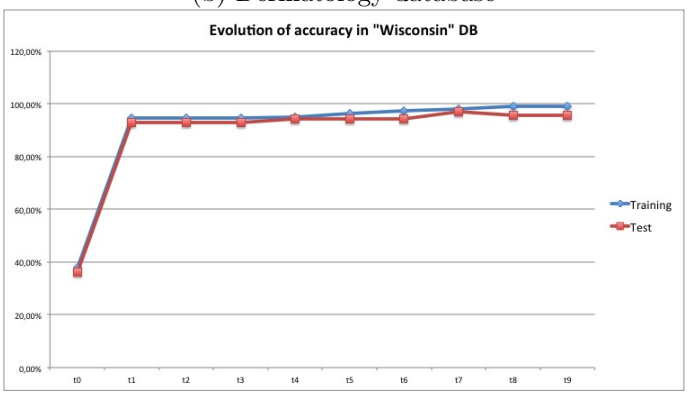
In general, it is not easy to identify this moment in the learning process and it is highly dependent of the problem concerned.



(a) Australian database



(b) Dermatology database



(c) Wisconsin database

Figure 6.7: Learning curve of SLAVE3* with the Australian, Dermatology and Wisconsin databases using 10 increments.

6.2.3.2.2 Second experiment

The second part of the experimental study focuses in the comparison between the incremental behavior of SLAVE3* and the non incremental version of SLAVE3* (equivalent to NSLV-AR), which means that the Wilcoxon's test has been applied. To distinguish among them, from now on, we will call SLAVE3* to the incremental model and NSLV-AR to the non-incremental version of SLAVE3*. Both methods use the default settings and the incremental approach makes the whole process in two steps. It starts learning with the 90% of the training examples and after that, in a second step, it uses the remaining 10%.

We have globally centred the analysis in the study of four parameters: accuracy on training and testing sets, the average number of rules and the average number of conditions per rule base.

Table 6.23: Results obtained by the Wilcoxon's test for SLAVE3* (accuracy on training set), using $\alpha = 0.05$.

Vs	R^+	R^-	p-value	Hypothesis
NSLV-AR	573.0	207.0	0.009155	**Rejected**

Table 6.24: Results obtained by the Wilcoxon's test for SLAVE3* (accuracy on testing set), using $\alpha = 0.05$.

Vs	R^+	R^-	p-value	Hypothesis
NSLV-AR	462.5	357.5	0.46796	**Not Rejected**

Table 6.25: Results obtained by the Wilcoxon's test for NSLV-AR (average number of rules), using $\alpha = 0.05$.

Vs	R^+	R^-	p-value	Hypothesis
SLAVE3*	697.0	123.0	0.000074	**Rejected**

Table 6.26: Results obtained by the Wilcoxon's test for NSLV-AR (time to obtain the model), using $\alpha = 0.05$.

Vs	R^+	R^-	p-value	Hypothesis
SLAVE3*	594.5	225.5	0.011862	**Rejected**

According to the results shown in tables 6.23, 6.24 and B.18, we can see that SLAVE3* obtains better results in accuracy, both in training and testing sets, achieving significant differences in the first parameter, where the null hypothesis is rejected (Table 6.23). Looking at the average values we realise that both models get similar results (specially in testing set), so we can finally conclude that this incremental proposal does not loose prediction

capability despite the use of incremental example sets. Moreover, it slightly improves the non-incremental model.

Regarding to the other two parameters (tables 6.25, 6.26 and B.19), we observe that the number of rules is significantly increased in the proposed incremental model. Looking at each single dataset, we can see that the difference is very small in most cases and the average results are strongly influenced by the *ring* (which almost triples the number of rules obtained by NSLV-AR), and *vehicle* datasets. This higher number of rules could be due, however, to the isolation of each set of examples considered in each iteration (characterized by the incremental approach).

Talking about the time parameter, the null hypothesis is again rejected (Table 6.26), so NSLV-AR presents the best performance in this case. Anyway, this conclusion is strongly conditioned by the results achieved in the *ring* dataset (where the time invested by SLAVE3* is more than three times the time employed by the original one). This parameter measures (in seconds), the time needed to obtain the final rule model. This means that in NSLV-AR, it measures the time needed to obtain the model considering the 100% of training examples. On the other hand, the time in SLAVE3* includes the two previously mentioned steps (learning using the 90% of training examples and, after that, the remaining 10%). It is important to point out this event, because one of the most interesting conclusions comes from the fact that the time needed for learning the last set of examples (this information has not been shown in order not to overload the tables), is less than the time needed to learn with the non-incremental version.

Chapter 7

Conclusions and Future Work

It is a fact that nowadays is ever more frequent to find problems working with large amounts of data. These data frequently include vague or imprecise information, difficult to handle. Due to this complexity, learning techniques must deal with these difficulties and evolve in order to represent in an accurate and efficient way these problems.

Inside this framework, the main purpose in this dissertation is basically to consider the indirect relevance of the input attributes in the learning process and also to improve the IRL model used in NSLV to be able to iteratively review the learned knowledge when working with a wide variety of data.

In this sense, the first objective is achieved through the use of feature construction techniques which allows the extraction of additional information resulting from the combination between the original variables. So, the learning algorithm handles not only the information given by the input variables, but also that given by the new constructed features. Following this idea, in this work we have presented three methods including feature construction techniques: one of them using relations in the antecedent of fuzzy rules, another one using functions in the antecedent of such rules and the last one combining both relations and functions.

The experimental results demonstrate a significant improvement of the prediction capability of the proposals using feature construction with regard to NSLV. Moreover, when comparing among them and talking about average values, NSLV-FR obtains the best results in accuracy closely followed by

NSLV-F and NSLV-R. The penalty here, comes from the other two parameters: the average number of rules and the time needed to get the model. According to the results, NSLV is clearly better than the rest on average, something expected at all.

So, it was experimentally proved that feature construction techniques work well when looking for accurate models, with an interpretability level nearer to natural language. Nevertheless, some well-known interpretability measures refers to the number of rules of rule bases and the number of conditions per rule (also per rule base), as key elements in order to consider a ruleset as interpretable.

It is inside this searching process of interpretable rule bases where the second main objective takes place. The ability of a fuzzy rule-based learning algorithm to review the knowledge as part of its own learning process, allows tuning the knowledge in each step. In this way, our proposal including knowledge review (NSLV-AR), decides in each iteration whether to replace or not a previously learned set of rules.

From the experimental point of view, NSLV-AR definitely demonstrates having a good performance, significantly reducing the complexity of both, rules and rulesets, without loosing prediction capability.

Finally, the last proposal (SLAVE3), arises from the need to integrate the ideas previously exposed into a new model achieving a good trade-off between accuracy and interpretability. So, on the one hand we were looking for an algorithm with a high level of accuracy, similar to those using feature construction, and on the other hand, which also were able to improve the interpretability (by reducing the complexity at a rule/rule base level), when compared with those last ones.

According to the experimental results, SLAVE3 is close enough in average to NSLV-FR to be considered an accurate algorithm (following the expected behavior of an algorithm using feature construction), but significantly reduces the number of conditions per rule regarding this last one (with a significance level of 0.052). Another important point is that the average number of rules is also reduced when compared with NSLV-FR.

When comparing SLAVE3 against other learning algorithms, its behavior is quite similar to the one previously described. It maintains a high level of accuracy but achieving simpler rulesets. In brief, we may conclude that SLAVE3 finds a good trade-off among accuracy and interpretability at a rule base level when working with different real-world problems.

Concerning to future work, next steps are guided to improve the configuration capabilities of SLAVE3 as well as its adaptation in order to be included into the KEEL platform. In the same way, the idea is to extend some of the methodologies here exposed with the purpose of working on Ordinal Classification Problems (OCM). These kind of problems are mainly characterized by working with a set of ordered classes with the objective of minimizing the error between the known class value and the class value

predicted by the learning algorithm.

Last, one more idea would be related to the development of the incremental model briefly described in this dissertation towards an on-line learning algorithm. Thus, the problem of learning new concepts (not initially included in the system knowledge base), while the system goes on working, should be handled. In this sense, two main tasks should be carried out. The first one is more related to the detection of the degradation level of the knowledge base while the second one would be responsible for learning the examples obtained from the interaction with the environment.

Chapter A

Experimental Settings

All the experimental results shown in this dissertation have been obtained through the KEEL platform [3] using 40 datasets available for this tool, with the same partitions in all databases and with the recommended settings. Table A.1 describes these databases, where each row represents the number of examples (#E), variables (#V), real variables (#R), integer variables (#I), nominal variables (#N), classes (#Cl), and missing values (M) respectively.

For all databases we have considered five uniformly distributed linguistic labels to define the domain of the continuous variables. The results have been obtained using ten-fold cross validation and the same partitions for all the algorithms.

To perform the comparison we used the nonparametric tests of Wilcoxon [101, 118] (for pairwise comparisons) and Friedman [31, 30] (for multiple comparisons, 1xN and NxN). All hypotheses of equality between the control method an the rest of algorithms (1xN), have been tested through the Holm's post-hoc procedure [57]. When necessary, in NxN comparisons, the hypotheses of equality between all existing pairs have been tested through the Shaffer's post-hoc procedure [98]. All comparisons have been performed by using a level of significance $\alpha = \mathbf{0.05}$.

Additionally, some chapters show examples of rule bases obtained over the *wdbc* database. This database, which is available in the KEEL platform [3], is fully described in Figure A.1.

Particularly, all these examples of rule bases have been extracted from the execution of the different algorithms over the 9^{th} partition (considering the partitions given in the KEEL platform when using ten-fold cross vali-

Table A.1: Datasets used in the experimental studies.

Dataset	#E	#V	#R	#I	#N	#Cl	M
Appendicitis	106	7	7	0	0	2	No
Australian	690	14	3	5	6	2	No
Automobile	205	25	15	0	10	6	Yes
Balance	625	4	4	0	0	3	No
Banana	5300	2	2	0	0	2	No
Breast	286	9	0	0	9	2	Yes
Bupa	345	6	1	5	0	2	No
Chess	3196	36	0	0	36	2	No
Cleveland	303	13	13	0	0	5	Yes
Contraceptive Method	1473	9	0	9	0	3	No
Credit Approval	690	15	3	3	9	2	Yes
Dermatology	366	34	0	34	0	6	Yes
Ecoli	336	7	7	0	0	8	No
Flare	1066	11	0	0	11	6	No
Glass	214	9	9	0	0	7	No
Haberman	306	3	0	3	0	2	No
Hayes-Roth	160	4	0	4	0	3	No
Heart	270	13	1	12	0	2	No
Housevotes	435	16	0	0	16	2	Yes
Iris	150	4	4	0	0	3	No
Led7Digit	500	7	7	0	0	10	No
Lymphography	148	18	0	3	15	4	No
Monk-2	432	6	0	6	0	2	No
Movement Libras	360	90	90	0	0	15	No
New Thyroid	215	5	4	1	0	3	No
Pima	768	8	8	0	0	2	No
Post-operative	87	8	0	0	8	3	Yes
Ring	7400	20	20	0	0	2	No
Saheart	462	9	5	3	1	2	No
Sonar	208	60	60	0	0	2	No
Spectfheart	267	44	0	44	0	2	No
Splice	3190	60	0	0	60	3	No
Thyroid	7200	21	6	15	0	3	No
Titanic	2201	3	3	0	0	2	No
Vehicle	846	18	0	18	0	4	No
Vowel	990	13	10	3	0	11	No
Wisconsing Diagnostic	569	30	30	0	0	2	No
Wine	178	13	13	0	0	3	No
Wisconsing	699	9	0	9	0	2	Yes
Yeast	1484	8	8	0	0	10	No

The "Breast Cancer Wisconsin (Diagnostic)" (wdbc) database contains 30 features computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.
The task is to determine if a found tumor is benign or malignant (M = malignant, B = benign).
The ten real-valued features that are computed for each of three different cell nucleus are the following:

1. radius: Mean of distances from center to points on the perimeter.
2. texture: Standard deviation of gray-scale values.
3. Perimeter.
4. Area.
5. Smoothness: local variation in radius lengths.
6. Compactness: $perimeter^2/area - 1.0$.
7. Concavity: severity of concave portions of the contour.
8. Concave points: number of concave portions of the contour.
9. Symmetry.
10. Fractal dimension: "coastline approximation" - 1.

Figure A.1: Description of the *wdbc* database.

dation), of the *wdbc* database. Taking under consideration the information given in Figure A.1, we can see that this dataset has two classes (M, B), whose distribution of examples in the 9^{th} partition is given below:

- Class M = 191 examples in training set and 21 examples in testing set.
- Class B = 321 examples in training set and 36 examples in testing set.

Chapter B

Comparative Tables

In this chapter we will present the comparative tables resulting from the experimental analysis carried out with the purpose of demonstrating the performance of the methods described in this dissertation. Thus, the tables are ordered according to the chapter they belong to.

B.1 Chapter 3: results obtained by SLAVE, SLAVE2, and NSLV on 40 databases

Table B.1: Results obtained by SLAVE, SLAVE2, and NSLV on 40 databases. The table shows the accuracy on training set.

Data	Training		
	SLAVE	SLAVE2	NSLV
appendicitis	91.4 (3)	91.7 (2)	**93.3 (1)**
australian	88.3 (3)	89.1 (2)	**90.3 (1)**
automobile	**99.5 (1)**	98.3 (2)	97.6 (3)
balance	93.8 (2)	**94.4 (1)**	85.8 (3)
banana	**78 (1)**	77 (2)	76.5 (3)
breast	**93.4 (1)**	92.1 (2)	86 (3)
bupa	64.9 (3)	65.5 (2)	**69.9 (1)**
chess	97.9 (2)	**98.4 (1)**	95 (3)
cleveland	**94.3 (1)**	91.6 (3)	92.9 (2)
contraceptive	**62.7 (1)**	59.6 (2)	49.3 (3)
crx	94 (2)	**95.2 (1)**	89.9 (3)
dermatology	**99.9 (1)**	99.6 (2)	99.1 (3)
ecoli	**88.4 (1)**	87.8 (2)	87.3 (3)
flare	**76.8 (1)**	76.7 (2)	74 (3)
glass	71 (2)	70.6 (3)	**80.2 (1)**
haberman	77.9 (2)	77.4 (3)	**78.1 (1)**
hayes-roth	**89.7 (1)**	89.3 (2)	88.7 (3)
heart	**96.8 (1)**	95.3 (2)	91.7 (3)
housevotes	**100 (1)**	99.9 (2)	97.4 (3)
iris	97.2 (3)	**97.3 (1.5)**	**97.3 (1.5)**
led7digit	**71.1 (1)**	69 (2)	65.2 (3)
lymphography	**98.7 (1)**	98.3 (2)	96.3 (3)
monk-2	**100 (1.5)**	**100 (1.5)**	97.5 (3)
movement libras	91.4 (3)	97.2 (2)	**98.2 (1)**
new thyroid	93.3 (2)	93.2 (3)	**95.1 (1)**
pima	78.4 (2)	77.9 (3)	**79.5 (1)**
post operative	**92.8 (1)**	90.3 (2)	81.6 (3)
ring	91.8 (3)	93.4 (2)	**94.9 (1)**
saheart	79.9 (2)	78.2 (3)	**81.1 (1)**
sonar	**99.4 (1)**	97.9 (3)	98.1 (2)
spectfheart	81.5 (2)	**81.9 (1)**	79.8 (3)
splice	59.3 (3)	**98.3 (1)**	94.6 (2)
thyroid	92.6 (3)	**93.1 (1)**	93 (2)
titanic	**79 (1.5)**	**79 (1.5)**	77.2 (3)
vehicle	70.3 (3)	70.4 (2)	**84.5 (1)**
vowel	77.9 (3)	79.1 (2)	**94.5 (1)**
wdbc	95.2 (3)	96.5 (2)	**98.1 (1)**
wine	**99.8 (1.5)**	99.7 (3)	**99.8 (1.5)**
wisconsin	**99.6 (1)**	99.3 (2)	98.3 (3)
yeast	**56.8 (1)**	43.6 (3)	51.8 (2)
Mean	86.6175	87.0775	86.985

B.1. Chapter 3: results obtained by SLAVE, SLAVE2, and NSLV on 40 databases

Table B.2: Results obtained by SLAVE, SLAVE2, and NSLV on 40 databases. The table shows the accuracy on testing set.

Data	Test		
	SLAVE	SLAVE2	NSLV
appendicitis	84.6 (3)	**86.4 (1)**	85.4 (2)
australian	83 (2.5)	83 (2.5)	**85 (1)**
automobile	77.1 (2)	**79.4 (1)**	74.2 (3)
balance	64.4 (3)	66.9 (2)	**76.3 (1)**
banana	**78 (1)**	77 (2)	76.5 (3)
breast	65.4 (2)	64.8 (3)	**72.9 (1)**
bupa	**61.6 (1)**	59.6 (2)	57.9 (3)
chess	97.4 (2)	**98.1 (1)**	94.6 (3)
cleveland	49.5 (3)	**52.4 (1)**	50.1 (2)
contraceptive	**43.7 (1)**	41.3 (2)	35.7 (3)
crx	83.3 (2.5)	83.3 (2.5)	**84.6 (1)**
dermatology	85.9 (3)	92.6 (2)	**96.6 (1)**
ecoli	82.4 (2)	**82.7 (1)**	80.4 (3)
flare	67.9 (2)	**68.2 (1)**	67 (3)
glass	59.5 (3)	59.8 (2)	**61.7 (1)**
haberman	71.8 (2.5)	71.8 (2.5)	**72.8 (1)**
hayes-roth	76.8 (2)	75.6 (3)	**78.1 (1)**
heart	73.3 (3)	78.8 (2)	**80 (1)**
housevotes	**96.5 (1)**	95.7 (3)	96 (2)
iris	96 (2.5)	96 (2.5)	**97.3 (1)**
led7digit	**63.6 (1)**	62 (2)	59.9 (3)
lymphography	67.7 (3)	68.2 (2)	**77.1 (1)**
monk-2	**100 (1.5)**	**100 (1.5)**	97.5 (3)
movement libras	70 (3)	79.9 (2)	**80.2 (1)**
new thyroid	91.6 (2.5)	91.6 (2.5)	**92.6 (1)**
pima	**74.7 (1)**	73.1 (3)	74.4 (2)
post operative	56.2 (2)	50.8 (3)	**64.4 (1)**
ring	91.8 (3)	**93 (1)**	92.9 (2)
saheart	**69.2 (1.5)**	69 (3)	**69.2 (1.5)**
sonar	67.7 (3)	**80.3 (1)**	76.9 (2)
spectfheart	78.6 (3)	**79.4 (1.5)**	**79.4 (1.5)**
splice	7.7 (3)	90 (2)	**91.1 (1)**
thyroid	92.6 (3)	**93 (1.5)**	**93 (1.5)**
titanic	**78.8 (1.5)**	**78.8 (1.5)**	76.8 (3)
vehicle	62.9 (3)	64.6 (2)	**67.8 (1)**
vowel	74.4 (2)	72.8 (3)	**82 (1)**
wdbc	93.1 (3)	**95 (1)**	94.5 (2)
wine	96 (2)	**96.6 (1)**	93.2 (3)
wisconsin	92.8 (3)	92.9 (2)	**94.4 (1)**
yeast	**54.9 (1)**	41 (3)	49.1 (2)
Mean	74.56	77.135	78.2375

Table B.3: Results obtained by SLAVE, SLAVE2, and NSLV on 40 databases. The table shows the average number of rules.

Data	Rules		
	SLAVE	SLAVE2	NSLV
appendicitis	6.2 (3)	5.6 (2)	**5.3 (1)**
australian	**12.4 (1.5)**	15.6 (3)	**12.4 (1.5)**
automobile	27.7 (3)	22.8 (2)	**18.2 (1)**
balance	72.1 (3)	70.7 (2)	**17.8 (1)**
banana	8 (3)	7.9 (2)	**6.3 (1)**
breast	19.4 (3)	18.8 (2)	**12.3 (1)**
bupa	8.3 (2)	9.8 (3)	**6.7 (1)**
chess	12.7 (2)	13 (3)	**5.3 (1)**
cleveland	57.7 (3)	53.7 (2)	**50.9 (1)**
contraceptive	74 (3)	60.5 (2)	**44.8 (1)**
crx	21.3 (2)	23.2 (3)	**7.7 (1)**
dermatology	23 (3)	12 (2)	**9.7 (1)**
ecoli	19 (3)	16 (2)	**14.6 (1)**
flare	**26.5 (1)**	26.6 (2)	29.1 (3)
glass	18 (3)	17.6 (2)	**15.1 (1)**
haberman	8.8 (3)	7.6 (2)	**4.8 (1)**
hayes-roth	10 (3)	9.8 (2)	**9 (1)**
heart	21.5 (3)	19.7 (2)	**11.2 (1)**
housevotes	6.2 (3)	5.9 (2)	**2.7 (1)**
iris	6 (3)	5.9 (2)	**3.5 (1)**
led7digit	19.3 (3)	15.7 (2)	**12.3 (1)**
lymphography	13.7 (3)	12.9 (2)	**10.4 (1)**
monk-2	4 (2.5)	4 (2.5)	**2.5 (1)**
movement libras	112.4 (3)	91.1 (2)	**49.9 (1)**
new thyroid	9.2 (3)	7.6 (2)	**6 (1)**
pima	13.7 (3)	12.5 (2)	**12 (1)**
post operative	11.6 (3)	10 (2)	**5.7 (1)**
ring	**12.2 (1)**	14.3 (2)	29.3 (3)
saheart	25.4 (3)	18.5 (2)	**14.7 (1)**
sonar	59.9 (3)	21.4 (2)	**10.8 (1)**
spectfheart	11.8 (2)	12.5 (3)	**2.1 (1)**
splice	1529.8 (3)	58.9 (2)	**15.5 (1)**
thyroid	4 (2)	5.9 (3)	**3.2 (1)**
titanic	4.1 (2.5)	4.1 (2.5)	**3.7 (1)**
vehicle	23.5 (2)	**22.1 (1)**	41.1 (3)
vowel	82.2 (2)	**76.8 (1)**	82.3 (3)
wdbc	**7.5 (1)**	11.5 (3)	8.1 (2)
wine	13 (2)	13.6 (3)	**6.5 (1)**
wisconsin	11 (3)	9.9 (2)	**8 (1)**
yeast	24.3 (3)	**16.1 (1)**	19.9 (2)
Mean	61.285	21.5525	15.785

B.1. Chapter 3: results obtained by SLAVE, SLAVE2, and NSLV on 40 databases

Table B.4: Results obtained by SLAVE, SLAVE2, and NSLV on 40 databases. The table shows the time employed to get the model (in seconds).

Data	Time		
	SLAVE	SLAVE2	NSLV
appendicitis	1 (2)	1.5 (3)	**0.6 (1)**
australian	11.6 (2)	21.9 (3)	**6.2 (1)**
automobile	10.6 (2)	13 (3)	**3.9 (1)**
balance	23.4 (2)	45.3 (3)	**3.8 (1)**
banana	24.8 (2)	47.5 (3)	**9.3 (1)**
breast	6.2 (2)	9.8 (3)	**2 (1)**
bupa	3.9 (2)	7.5 (3)	**1.8 (1)**
chess	108 (3)	91.6 (2)	**12.8 (1)**
cleveland	31.9 (2)	43.6 (3)	**18.9 (1)**
contraceptive	111.6 (2)	167.5 (3)	**51.3 (1)**
crx	18.9 (2)	29.3 (3)	**4.5 (1)**
dermatology	20.5 (3)	11.7 (2)	**3.4 (1)**
ecoli	8.1 (2)	12.5 (3)	**2.7 (1)**
flare	28.3 (2)	57.4 (3)	**16.5 (1)**
glass	7.3 (2)	11.1 (3)	**2.7 (1)**
haberman	2 (2)	3.1 (3)	**0.7 (1)**
hayes-roth	1.5 (2)	2.4 (3)	**0.7 (1)**
heart	8.2 (2)	10.3 (3)	**2.4 (1)**
housevotes	1.7 (3)	1.6 (2)	**0.4 (1)**
iris	1 (2)	1.3 (3)	**0.3 (1)**
led7digit	11.9 (2)	17.6 (3)	**1.9 (1)**
lymphography	3.8 (2)	3.9 (3)	**1.2 (1)**
monk-2	1.7 (2.5)	1.7 (2.5)	**0.4 (1)**
movement libras	391.1 (2)	410.2 (3)	**110 (1)**
new thyroid	2.1 (2)	2.8 (3)	**0.7 (1)**
pima	13.8 (2)	21.6 (3)	**7.1 (1)**
post operative	1.1 (2)	1.7 (3)	**0.4 (1)**
ring	547.3 (2)	**393.2 (1)**	590.5 (3)
saheart	16.1 (2)	17.3 (3)	**5.5 (1)**
sonar	57 (3)	35.7 (2)	**7.9 (1)**
spectfheart	31.8 (2)	38.2 (3)	**2 (1)**
splice	9343.6 (3)	726 (2)	**61.7 (1)**
thyroid	62.6 (2)	109.3 (3)	**14 (1)**
titanic	8.2 (2)	12.9 (3)	**2 (1)**
vehicle	77.3 (2)	97 (3)	**58.7 (1)**
vowel	**138.3 (1)**	221.3 (3)	138.6 (2)
wdbc	26.3 (2)	30.7 (3)	**9.7 (1)**
wine	4.9 (2)	6 (3)	**1.3 (1)**
wisconsin	9 (2)	12.5 (3)	**4.1 (1)**
yeast	55.8 (2)	68.8 (3)	**15.2 (1)**
Mean	280.855	70.4575	29.445

B.2 Chapter 4: results obtained by NSLV, NSLV-R, NSLV-F and NSLV-FR on 40 databases

Table B.5: Results obtained by NSLV, NSLV-R, NSLV-F and NSLV-FR on 40 databases. The table shows the accuracy on training set.

Data	Training			
	NSLV	NSLV-R	NSLV-F	NSLV-FR
appendicitis	**93.3 (1)**	92.8 (3)	93.2 (2)	92 (4)
australian	90.3 (2)	88.7 (4)	89.5 (3)	**90.7 (1)**
automobile	97.6 (4)	98.5 (3)	**99.3 (1)**	99.1 (2)
balance	85.8 (4)	98.1 (2)	94.6 (3)	**98.4 (1)**
banana	76.5 (4)	79.3 (3)	87 (2)	**88.8 (1)**
breast	**86 (1)**	81 (3)	79.9 (4)	82.4 (2)
bupa	69.9 (4)	78.8 (3)	80.7 (2)	**85.3 (1)**
chess	**95 (1)**	94.3 (4)	94.5 (2.5)	94.5 (2.5)
cleveland	**92.9 (1)**	86.3 (4)	89.4 (3)	90.6 (2)
contraceptive	49.3 (4)	63.7 (3)	**70.3 (1)**	70.1 (2)
crx	**89.9 (1)**	89.5 (2)	87.5 (4)	88.9 (3)
dermatology	99.1 (4)	**99.6 (1.5)**	99.5 (3)	**99.6 (1.5)**
ecoli	87.3 (4)	94.1 (2)	92.8 (3)	**95.2 (1)**
flare	74 (4)	**80.7 (1)**	79.3 (3)	80.4 (2)
glass	80.2 (4)	83.6 (2)	83.3 (3)	**85.6 (1)**
haberman	**78.1 (1)**	73.7 (4)	74.6 (3)	74.7 (2)
hayes-roth	88.7 (3)	88.1 (4)	**90.9 (1)**	90.6 (2)
heart	91.7 (4)	92.3 (2)	91.9 (3)	**94.4 (1)**
housevotes	97.4 (4)	97.7 (3)	97.8 (2)	**98.3 (1)**
iris	97.3 (3)	97 (4)	**99.1 (1)**	98.9 (2)
led7digit	65.2 (4)	79 (3)	**79.2 (1.5)**	**79.2 (1.5)**
lymphography	96.3 (2)	**96.6 (1)**	95.1 (4)	96.1 (3)
monk-2	97.5 (4)	**99.1 (1)**	98.6 (3)	98.7 (2)
movement libras	98.2 (4)	99.2 (3)	**99.5 (1)**	99.4 (2)
new thyroid	95.1 (4)	96.4 (3)	**97.6 (1.5)**	**97.6 (1.5)**
pima	79.5 (4)	81.8 (3)	82 (2)	**82.8 (1)**
post operative	**81.6 (1)**	76.7 (3)	74 (4)	78.4 (2)
ring	94.9 (4)	95.1 (3)	**97.2 (1)**	96.1 (2)
saheart	81.1 (3)	81 (4)	82.2 (2)	**85.1 (1)**
sonar	98.1 (2)	97.6 (4)	98 (3)	**98.7 (1)**
spectfheart	79.8 (2)	79.7 (3)	79.4 (4)	**80.5 (1)**
splice	94.6 (3)	94.9 (2)	94.4 (4)	**96 (1)**
thyroid	93 (3)	93.8 (2)	92.7 (4)	**94 (1)**
titanic	77.2 (4)	78.3 (3)	**78.6 (1)**	78.4 (2)
vehicle	84.5 (4)	87.1 (3)	**90 (1)**	89.1 (2)
vowel	94.5 (4)	94.8 (3)	**97 (1)**	96.1 (2)
wdbc	98.1 (4)	98.7 (2.5)	**98.8 (1)**	98.7 (2.5)
wine	99.8 (4)	**100 (2)**	**100 (2)**	**100 (2)**
wisconsin	98.3 (4)	98.7 (2.5)	98.7 (2.5)	**98.8 (1)**
yeast	51.8 (4)	65.3 (3)	67.9 (2)	**70.1 (1)**
Mean	86.985	88.79	89.4	90.3075

B.2. Chapter 4: results obtained by NSLV, NSLV-R, NSLV-F and NSLV-FR on 40 databases

Table B.6: Results obtained by NSLV, NSLV-R, NSLV-F and NSLV-FR on 40 databases. The table shows the accuracy on testing set.

Data	Test			
	NSLV	NSLV-R	NSLV-F	NSLV-FR
appendicitis	85.4 (2)	**85.5 (1)**	84.5 (3)	84.4 (4)
australian	85 (4)	85.6 (2.5)	**85.7 (1)**	85.6 (2.5)
automobile	74.2 (4)	80.6 (2)	78.4 (3)	**81.3 (1)**
balance	76.3 (4)	**98.1 (1)**	89 (3)	96.2 (2)
banana	76.5 (4)	78.6 (3)	86.5 (2)	**88 (1)**
breast	72.9 (2)	70.7 (4)	**73.3 (1)**	72.6 (3)
bupa	57.9 (4)	**70.4 (1)**	67.1 (2)	66.4 (3)
chess	**94.6 (1.5)**	94.2 (3.5)	94.2 (3.5)	**94.6 (1.5)**
cleveland	50.1 (4)	**56.1 (1)**	54.7 (3)	55 (2)
contraceptive	35.7 (4)	52.3 (2.5)	**55.1 (1)**	52.3 (2.5)
crx	84.6 (4)	86 (2)	**86.4 (1)**	85.2 (3)
dermatology	**96.6 (1)**	93.5 (4)	95.2 (2.5)	95.2 (2.5)
ecoli	80.4 (2)	**81.5 (1)**	80 (3)	79.7 (4)
flare	67 (4)	74.8 (2)	**75 (1)**	74.3 (3)
glass	61.7 (4)	**69.9 (1)**	64.9 (3)	68.8 (2)
haberman	72.8 (2.5)	72.8 (2.5)	**73.1 (1)**	72.2 (4)
hayes-roth	78.1 (3)	**80.6 (1)**	79.3 (2)	77.5 (4)
heart	80 (3)	80.3 (2)	**84.4 (1)**	79.6 (4)
housevotes	96 (4)	96.2 (2.5)	96.2 (2.5)	**97.1 (1)**
iris	**97.3 (1.5)**	**97.3 (1.5)**	94.6 (4)	95.3 (3)
led7digit	59.9 (4)	**72 (1)**	71.6 (2)	71.2 (3)
lymphography	77.1 (3)	76.8 (4)	78.3 (2)	**79.8 (1)**
monk-2	97.5 (4)	**99 (1)**	97.9 (2.5)	97.9 (2.5)
movement libras	**80.2 (1)**	74.7 (3)	74.1 (4)	76.1 (2)
new thyroid	92.6 (3)	91.6 (4)	**93.5 (1)**	93 (2)
pima	74.4 (2)	73.5 (3)	**74.8 (1)**	72.9 (4)
post operative	64.4 (4)	67.9 (2)	**70.1 (1)**	64.5 (3)
ring	92.9 (2)	91.8 (4)	**93.6 (1)**	92.8 (3)
saheart	69.2 (3)	70.5 (2)	**71 (1)**	68.4 (4)
sonar	76.9 (3)	77.2 (2)	72 (4)	**79.3 (1)**
spectfheart	79.4 (3)	**79.7 (1)**	79.4 (3)	79.4 (3)
splice	91.1 (4)	93.5 (3)	93.7 (2)	**94.2 (1)**
thyroid	93 (3)	93.7 (2)	92.7 (4)	**93.9 (1)**
titanic	76.8 (4)	78.3 (2.5)	**78.6 (1)**	78.3 (2.5)
vehicle	67.8 (4)	69.4 (3)	70.5 (2)	**73.1 (1)**
vowel	82 (2)	77.4 (4)	**82.1 (1)**	79.9 (3)
wdbc	94.5 (3)	94.1 (4)	**94.7 (1.5)**	**94.7 (1.5)**
wine	93.2 (3.5)	**94.9 (1)**	93.2 (3.5)	93.8 (2)
wisconsin	94.4 (4)	95.1 (3)	**95.8 (1)**	95.4 (2)
yeast	49.1 (4)	54.9 (3)	**57.6 (1)**	56.8 (2)
Mean	78.2375	80.775	80.82	80.9175

Table B.7: Results obtained by NSLV, NSLV-R, NSLV-F and NSLV-FR on 40 databases. The table shows the average number of rules.

Data	Rules			
	NSLV	NSLV-R	NSLV-F	NSLV-FR
appendicitis	5.3 (4)	4.8 (3)	4.5 (2)	**4.2 (1)**
australian	12.4 (4)	**7.1 (1)**	8.7 (2)	10.7 (3)
automobile	18.2 (2.5)	**18 (1)**	18.4 (4)	18.2 (2.5)
balance	17.8 (3)	**9 (1)**	18.2 (4)	12.8 (2)
banana	**6.3 (1)**	10.3 (2)	16 (3)	22.3 (4)
breast	12.3 (4)	6.1 (2)	**6 (1)**	6.7 (3)
bupa	**6.7 (1)**	11.9 (2)	14.5 (3)	20.4 (4)
chess	5.3 (4)	**4.1 (1)**	4.7 (3)	4.3 (2)
cleveland	50.9 (4)	**32.2 (1)**	35.2 (2)	37 (3)
contraceptive	**44.8 (1)**	54.1 (2)	76.4 (3)	80.1 (4)
crx	7.7 (4)	6 (3)	**4 (1)**	5.1 (2)
dermatology	9.7 (2)	11.6 (4)	**9.5 (1)**	10.3 (3)
ecoli	**14.6 (1)**	24.7 (3)	23 (2)	27.1 (4)
flare	29.1 (4)	24.9 (2)	**22.2 (1)**	25.6 (3)
glass	**15.1 (1)**	15.7 (2)	16.2 (3)	16.6 (4)
haberman	4.8 (4)	**2.1 (1)**	2.6 (2)	2.8 (3)
hayes-roth	9 (2.5)	9 (2.5)	**8.9 (1)**	9.1 (4)
heart	11.2 (3)	10.9 (2)	**9.7 (1)**	12.5 (4)
housevotes	**2.7 (1)**	3 (2)	3.2 (3)	3.3 (4)
iris	3.5 (2)	**3.1 (1)**	4.2 (3)	4.6 (4)
led7digit	**12.3 (1)**	21.7 (4)	21.3 (3)	20.6 (2)
lymphography	10.4 (4)	9.7 (3)	**8.5 (1)**	9.1 (2)
monk-2	**2.5 (1)**	3.7 (3)	3.4 (2)	4.1 (4)
movement libras	49.9 (3)	**47.8 (1)**	51.9 (4)	48.1 (2)
new thyroid	**6 (1.5)**	6.3 (3)	7 (4)	**6 (1.5)**
pima	**12 (1)**	17.6 (2)	19.1 (3)	23 (4)
post operative	5.7 (4)	3.5 (2)	**2.5 (1)**	3.8 (3)
ring	**29.3 (1)**	68.6 (2)	76.5 (4)	76.4 (3)
saheart	**14.7 (1)**	17.9 (2)	19.8 (3)	24.9 (4)
sonar	**10.8 (1)**	12 (2)	12.4 (4)	12.2 (3)
spectfheart	2.1 (2)	2.2 (3.5)	**2 (1)**	2.2 (3.5)
splice	15.5 (3)	15 (2)	**11.7 (1)**	20.9 (4)
thyroid	3.2 (3)	2.9 (2)	**2.5 (1)**	3.3 (4)
titanic	3.7 (4)	**2.3 (1.5)**	2.9 (3)	**2.3 (1.5)**
vehicle	**41.1 (1)**	53.9 (4)	53.4 (3)	52.6 (2)
vowel	82.3 (2)	93.3 (4)	**82.1 (1)**	86.4 (3)
wdbc	**8.1 (1)**	8.8 (3)	9 (4)	8.3 (2)
wine	**6.5 (1)**	7.4 (4)	6.7 (2)	7.2 (3)
wisconsin	8 (2.5)	8.4 (4)	**7.7 (1)**	8 (2.5)
yeast	**19.9 (1)**	56.5 (2)	64.7 (3)	76.2 (4)
Mean	15.785	18.2025	19.28	20.7325

B.2. Chapter 4: results obtained by NSLV, NSLV-R, NSLV-F and NSLV-FR on 40 databases

Table B.8: Results obtained by NSLV, NSLV-R, NSLV-F and NSLV-FR on 40 databases. The table shows the time employed to get the model (in seconds).

Data	Time			
	NSLV	NSLV-R	NSLV-F	NSLV-FR
appendicitis	**0.6 (1)**	0.7 (2)	1 (4)	0.9 (3)
australian	6.2 (4)	**2.9 (1)**	5.4 (2)	5.8 (3)
automobile	**3.9 (1)**	5.5 (2)	9 (4)	6.9 (3)
balance	3.8 (2)	**2.6 (1)**	10.7 (4)	5 (3)
banana	**9.3 (1)**	26.6 (2)	83 (3)	125.2 (4)
breast	2 (4)	**1 (1)**	1.4 (2)	1.5 (3)
bupa	**1.8 (1)**	3 (2)	6.1 (3)	7.3 (4)
chess	12.8 (3)	**8.4 (1)**	15.8 (4)	11.2 (2)
cleveland	18.9 (2)	**12.6 (1)**	30.7 (4)	24.7 (3)
contraceptive	**51.3 (1)**	141.7 (2)	255.5 (4)	243.4 (3)
crx	4.5 (4)	2.9 (3)	**1.7 (1)**	2.5 (2)
dermatology	**3.4 (1)**	4.1 (2)	8 (4)	5.4 (3)
ecoli	**2.7 (1)**	7.2 (2)	10.5 (3)	10.8 (4)
flare	**16.5 (1)**	22.8 (2)	32.5 (4)	26.5 (3)
glass	**2.7 (1)**	3.5 (2)	7.2 (4)	5.3 (3)
haberman	0.7 (4)	**0.4 (1)**	0.6 (2.5)	0.6 (2.5)
hayes-roth	**0.7 (1)**	0.8 (2)	1.2 (4)	1 (3)
heart	2.4 (2)	**2 (1)**	2.7 (3)	3.4 (4)
housevotes	**0.4 (1)**	0.5 (2.5)	0.5 (2.5)	0.6 (4)
iris	**0.3 (1)**	0.4 (2)	0.7 (3.5)	0.7 (3.5)
led7digit	**1.9 (1)**	7.6 (2)	15.4 (4)	12.2 (3)
lymphography	**1.2 (1.5)**	**1.2 (1.5)**	1.4 (4)	1.3 (3)
monk-2	**0.4 (1)**	0.8 (2.5)	0.8 (2.5)	0.9 (4)
movement libras	110 (2)	**95.9 (1)**	1158.7 (4)	388.6 (3)
new thyroid	**0.7 (1)**	0.9 (2)	1 (3)	1.2 (4)
pima	**7.1 (1)**	9.5 (2)	15.5 (3)	18.1 (4)
post operative	0.4 (3)	0.4 (3)	**0.3 (1)**	0.4 (3)
ring	**590.5 (1)**	692.4 (2)	1519.5 (4)	1239.9 (3)
saheart	**5.5 (1)**	6.5 (2)	12.3 (4)	12.2 (3)
sonar	7.9 (2)	**6.5 (1)**	12.5 (4)	10.8 (3)
spectfheart	2 (4)	**0.7 (1)**	1 (2.5)	1 (2.5)
splice	61.7 (3)	**53.3 (1)**	61.5 (2)	91.5 (4)
thyroid	14 (2)	**12.1 (1)**	15.3 (3)	19.8 (4)
titanic	**2 (1.5)**	**2 (1.5)**	2.8 (4)	2.4 (3)
vehicle	**58.7 (1)**	78.8 (2)	125.4 (4)	113.4 (3)
vowel	**138.6 (1)**	243 (2)	322.4 (4)	283.4 (3)
wdbc	9.7 (3)	**6.5 (1)**	9.9 (4)	8.3 (2)
wine	1.3 (2)	**1 (1)**	1.6 (4)	1.4 (3)
wisconsin	**4.1 (1)**	5 (2)	6.9 (3.5)	6.9 (3.5)
yeast	**15.2 (1)**	156.2 (2)	292.9 (3)	331.7 (4)
Mean	29.445	40.7475	101.533	75.8525

B.3 Chapter 5: Results obtained by NSLV and NSLV-AR on 40 databases

Table B.9: Results obtained by NSLV and NSLV-AR on 40 databases. The table shows the accuracy on training and testing sets.

Data	Training		Test	
	NSLV	NSLV-AR	NSLV	NSLV-AR
appendicitis	**93.3 (1)**	91.9 (2)	85.4 (2)	**87.1 (1)**
australian	**90.3 (1)**	87.4 (2)	85 (2)	**85.3 (1)**
automobile	**97.6 (1)**	97.3 (2)	74.2 (2)	**76 (1)**
balance	**85.8 (1)**	81.1 (2)	**76.3 (1)**	75.3 (2)
banana	**76.5 (1)**	75.2 (2)	**76.5 (1)**	74.6 (2)
breast	**86 (1)**	75.9 (2)	**72.9 (1)**	71.1 (2)
bupa	**69.9 (1)**	66.5 (2)	**57.9 (1)**	55.2 (2)
chess	**95 (1)**	94.4 (2)	**94.6 (1)**	94 (2)
cleveland	**92.9 (1)**	62.8 (2)	50.1 (2)	**56.1 (1)**
contraceptive	**49.3 (1)**	35.2 (2)	**35.7 (1)**	34.3 (2)
crx	**89.9 (1)**	86.2 (2)	84.6 (2)	**85.3 (1)**
dermatology	**99.1 (1.5)**	**99.1 (1.5)**	**96.6 (1)**	94.4 (2)
ecoli	**87.3 (1)**	83.6 (2)	**80.4 (1)**	77 (2)
flare	**74 (1)**	64.5 (2)	**67 (1)**	63.8 (2)
glass	**80.2 (1)**	74.9 (2)	61.7 (2)	**62.6 (1)**
haberman	**78.1 (1)**	76.7 (2)	72.8 (2)	**74.1 (1)**
hayes-roth	88.7 (2)	**90.6 (1)**	78.1 (2)	**80.6 (1)**
heart	**91.7 (1)**	90.8 (2)	**80 (1)**	79.6 (2)
housevotes	**97.4 (1)**	97 (2)	96 (2)	**97 (1)**
iris	**97.3 (1)**	96.5 (2)	**97.3 (1)**	95.3 (2)
led7digit	65.2 (2)	**70.5 (1)**	59.9 (2)	**66.3 (1)**
lymphography	**96.3 (1)**	95 (2)	77.1 (2)	**83.3 (1)**
monk-2	97.5 (2)	**98 (1)**	97.5 (2)	**97.9 (1)**
movement libras	**98.2 (1)**	96.7 (2)	**80.2 (1)**	72.5 (2)
new thyroid	**95.1 (1.5)**	**95.1 (1.5)**	**92.6 (1)**	92 (2)
pima	**79.5 (1)**	76.6 (2)	**74.4 (1)**	74.2 (2)
post operative	**81.6 (1)**	73.4 (2)	64.4 (2)	**70.1 (1)**
ring	**94.9 (1)**	92.1 (2)	**92.9 (1)**	91.5 (2)
saheart	**81.1 (1)**	76.5 (2)	69.2 (2)	**69.6 (1)**
sonar	98.1 (2)	**98.6 (1)**	**76.9 (1)**	74 (2)
spectfheart	**79.8 (1)**	79.4 (2)	**79.4 (1.5)**	**79.4 (1.5)**
splice	**94.6 (1)**	92.1 (2)	91.1 (2)	**91.7 (1)**
thyroid	**93 (1)**	92.8 (2)	**93 (1)**	92.8 (2)
titanic	**77.2 (1)**	72.2 (2)	**76.8 (1)**	72 (2)
vehicle	**84.5 (1)**	64.1 (2)	**67.8 (1)**	55.9 (2)
vowel	**94.5 (1)**	86.2 (2)	**82 (1)**	72.8 (2)
wdbc	**98.1 (1)**	98 (2)	94.5 (2)	**94.7 (1)**
wine	**99.8 (1)**	99.6 (2)	**93.2 (1)**	91.5 (2)
wisconsin	98.3 (2)	**98.4 (1)**	94.4 (2)	**94.7 (1)**
yeast	**51.8 (1)**	49.1 (2)	**49.1 (1)**	46.2 (2)
Mean	86.985	83.3	78.2375	77.545

B.3. Chapter 5: Results obtained by NSLV and NSLV-AR on 40 databases

Table B.10: Results obtained by NSLV and NSLV-AR on 40 databases. The table shows the average number of rules and the average number of conditions per rule base.

Data	Rules		Conditions	
	NSLV	NSLV-AR	NSLV	NSLV-AR
appendicitis	5.3 (2)	**3.7 (1)**	12.3 (2)	**7.4 (1)**
australian	12.4 (2)	**4.9 (1)**	49 (2)	**12.8 (1)**
automobile	18.2 (2)	**14.2 (1)**	70.2 (2)	**58 (1)**
balance	17.8 (2)	**10.7 (1)**	47.7 (2)	**19.3 (1)**
banana	6.3 (2)	**5.1 (1)**	12 (2)	**8.9 (1)**
breast	12.3 (2)	**4.6 (1)**	36.2 (2)	**9.3 (1)**
bupa	6.7 (2)	**4.3 (1)**	24.7 (2)	**13 (1)**
chess	5.3 (2)	**4.1 (1)**	16.7 (2)	**9.4 (1)**
cleveland	50.9 (2)	**8.5 (1)**	240.5 (2)	**34.2 (1)**
contraceptive	44.8 (2)	**3.2 (1)**	200.3 (2)	**10.8 (1)**
crx	7.7 (2)	**3.3 (1)**	25.7 (2)	**5.2 (1)**
dermatology	9.7 (2)	**8.9 (1)**	**32.7 (1)**	33.7 (2)
ecoli	14.6 (2)	**11.1 (1)**	51.5 (2)	**28.8 (1)**
flare	29.1 (2)	**4.4 (1)**	97.6 (2)	**9.9 (1)**
glass	15.1 (2)	**9.1 (1)**	52.2 (2)	**24.8 (1)**
haberman	4.8 (2)	**2.9 (1)**	10.3 (2)	**5.2 (1)**
hayes-roth	9 (2)	**8.4 (1)**	19.3 (2)	**13.2 (1)**
heart	11.2 (2)	**9.6 (1)**	36.1 (2)	**26.8 (1)**
housevotes	2.7 (2)	**2.2 (1)**	2.8 (2)	**1.4 (1)**
iris	3.5 (2)	**3 (1)**	4.2 (2)	**2.5 (1)**
led7digit	12.3 (2)	**11.4 (1)**	52.7 (2)	**36.4 (1)**
lymphography	10.4 (2)	**9.3 (1)**	27.6 (2)	**20.3 (1)**
monk-2	**2.5 (1)**	2.8 (2)	**2.3 (1)**	2.7 (2)
movement libras	49.9 (2)	**41.1 (1)**	345.3 (2)	**261 (1)**
new thyroid	**6 (1)**	6.1 (2)	14.6 (2)	**10.2 (1)**
pima	12 (2)	**4.6 (1)**	43 (2)	**10.6 (1)**
post operative	5.7 (2)	**2.5 (1)**	13.7 (2)	**4.5 (1)**
ring	29.3 (2)	**10.4 (1)**	200.7 (2)	**63.3 (1)**
saheart	14.7 (2)	**7.8 (1)**	58 (2)	**22.1 (1)**
sonar	10.8 (2)	**10.1 (1)**	**54.6 (1)**	56.5 (2)
spectfheart	2.1 (2)	**1 (1)**	17.4 (2)	**0 (1)**
splice	15.5 (2)	**5.2 (1)**	70.2 (2)	**16.1 (1)**
thyroid	3.2 (2)	**1.3 (1)**	9 (2)	**3.2 (1)**
titanic	3.7 (2)	**2.5 (1)**	6.3 (2)	**4.2 (1)**
vehicle	41.1 (2)	**16.6 (1)**	227.8 (2)	**72.1 (1)**
vowel	82.3 (2)	**60.4 (1)**	511.8 (2)	**310.9 (1)**
wdbc	8.1 (2)	**6.9 (1)**	33 (2)	**24.5 (1)**
wine	**6.5 (1.5)**	**6.5 (1.5)**	22.4 (2)	**15.5 (1)**
wisconsin	**8 (1)**	8.8 (2)	**17.2 (1)**	18.1 (2)
yeast	19.9 (2)	**9 (1)**	72.4 (2)	**34.1 (1)**
Mean	15.785	8.7625	71.05	33.0225

B.4 Chapter 6

B.4.1 Results obtained by NSLV, NSLV-AR, NSLV-FR and SLAVE3 on 40 databases

Table B.11: Results obtained by NSLV, NSLV-AR, NSLV-FR and SLAVE3 on 40 databases. The table shows the accuracy on training set.

Data	Training			
	NSLV	NSLV-AR	NSLV-FR	SLAVE3
appendicitis	**93.3 (1)**	91.9 (4)	92 (3)	92.5 (2)
australian	90.3 (2)	87.4 (4)	**90.7 (1)**	89.2 (3)
automobile	97.6 (3)	97.3 (4)	**99.1 (1)**	98.1 (2)
balance	85.8 (3)	81.1 (4)	**98.4 (1)**	96.6 (2)
banana	76.5 (3)	75.2 (4)	88.8 (2)	**89.2 (1)**
breast	**86 (1)**	75.9 (4)	82.4 (2)	78.2 (3)
bupa	69.9 (3)	66.5 (4)	**85.3 (1)**	81.5 (2)
chess	**95 (1)**	94.4 (3)	94.5 (2)	94.3 (4)
cleveland	**92.9 (1)**	62.8 (4)	90.6 (2)	86.6 (3)
contraceptive	49.3 (3)	35.2 (4)	**70.1 (1)**	67 (2)
crx	**89.9 (1)**	86.2 (4)	88.9 (2.5)	88.9 (2.5)
dermatology	99.1 (3.5)	99.1 (3.5)	**99.6 (1)**	99.3 (2)
ecoli	87.3 (3)	83.6 (4)	**95.2 (1)**	93.6 (2)
flare	74 (3)	64.5 (4)	**80.4 (1)**	79.4 (2)
glass	80.2 (3)	74.9 (4)	**85.6 (1)**	83.8 (2)
haberman	**78.1 (1)**	76.7 (2)	74.7 (3)	73.6 (4)
hayes-roth	88.7 (4)	**90.6 (1.5)**	**90.6 (1.5)**	90.3 (3)
heart	91.7 (3)	90.8 (4)	**94.4 (1)**	92.5 (2)
housevotes	97.4 (3)	97 (4)	**98.3 (1)**	98 (2)
iris	97.3 (3)	96.5 (4)	**98.9 (1)**	97.7 (2)
led7digit	65.2 (4)	70.5 (3)	**79.2 (1)**	79.1 (2)
lymphography	**96.3 (1)**	95 (3)	96.1 (2)	94.9 (4)
monk-2	97.5 (4)	98 (3)	**98.7 (1)**	98.4 (2)
movement libras	98.2 (2)	96.7 (4)	**99.4 (1)**	97.5 (3)
new thyroid	95.1 (3.5)	95.1 (3.5)	**97.6 (1.5)**	**97.6 (1.5)**
pima	79.5 (3)	76.6 (4)	**82.8 (1)**	82.3 (2)
post operative	**81.6 (1)**	73.4 (3.5)	78.4 (2)	73.4 (3.5)
ring	94.9 (3)	92.1 (4)	96.1 (2)	**96.4 (1)**
saheart	81.1 (3)	76.5 (4)	**85.1 (1)**	82.5 (2)
sonar	98.1 (3)	98.6 (2)	**98.7 (1)**	97.2 (4)
spectfheart	79.8 (3)	79.4 (4)	**80.5 (1)**	79.9 (2)
splice	94.6 (2)	92.1 (4)	**96 (1)**	94.5 (3)
thyroid	93 (3)	92.8 (4)	**94 (1)**	93.4 (2)
titanic	77.2 (3)	72.2 (4)	**78.4 (1)**	78.3 (2)
vehicle	84.5 (3)	64.1 (4)	**89.1 (1)**	88.4 (2)
vowel	94.5 (3)	86.2 (4)	**96.1 (1)**	94.6 (2)
wdbc	98.1 (3)	98 (4)	**98.7 (1)**	98.4 (2)
wine	99.8 (2.5)	99.6 (4)	**100 (1)**	99.8 (2.5)
wisconsin	98.3 (4)	98.4 (3)	**98.8 (1)**	98.7 (2)
yeast	51.8 (3)	49.1 (4)	**70.1 (1)**	68.3 (2)
Mean	86.985	83.3	90.3075	89.0975

B.4. Chapter 6

Table B.12: Results obtained by NSLV, NSLV-AR, NSLV-FR and SLAVE3 on 40 databases. The table shows the accuracy on testing set.

Data	Test			
	NSLV	NSLV-AR	NSLV-FR	SLAVE3
appendicitis	85.4 (2)	**87.1 (1)**	84.4 (3)	82.8 (4)
australian	85 (4)	85.3 (3)	85.6 (2)	**86 (1)**
automobile	74.2 (4)	76 (3)	**81.3 (1)**	79.7 (2)
balance	76.3 (3)	75.3 (4)	**96.2 (1)**	95.9 (2)
banana	76.5 (3)	74.6 (4)	88 (2)	**88.3 (1)**
breast	**72.9 (1)**	71.1 (3)	72.6 (2)	70.3 (4)
bupa	57.9 (3)	55.2 (4)	66.4 (2)	**67.9 (1)**
chess	**94.6 (1.5)**	94 (4)	**94.6 (1.5)**	94.3 (3)
cleveland	50.1 (4)	56.1 (2)	55 (3)	**56.7 (1)**
contraceptive	35.7 (3)	34.3 (4)	52.3 (2)	**53.1 (1)**
crx	84.6 (4)	85.3 (2)	85.2 (3)	**85.7 (1)**
dermatology	**96.6 (1)**	94.4 (3)	95.2 (2)	93.5 (4)
ecoli	**80.4 (1)**	77 (4)	79.7 (3)	80 (2)
flare	67 (3)	63.8 (4)	74.3 (2)	**74.9 (1)**
glass	61.7 (4)	62.6 (3)	**68.8 (1)**	66.8 (2)
haberman	72.8 (3)	**74.1 (1)**	72.2 (4)	73.8 (2)
hayes-roth	78.1 (3)	80.6 (2)	77.5 (4)	**81.8 (1)**
heart	**80 (1)**	79.6 (2.5)	79.6 (2.5)	77.7 (4)
housevotes	96 (3)	97 (2)	**97.1 (1)**	95.7 (4)
iris	**97.3 (1)**	95.3 (2.5)	95.3 (2.5)	94.6 (4)
led7digit	59.9 (4)	66.3 (3)	71.2 (2)	**71.6 (1)**
lymphography	77.1 (4)	**83.3 (1)**	79.8 (2)	79.2 (3)
monk-2	97.5 (3.5)	**97.9 (1.5)**	**97.9 (1.5)**	97.5 (3.5)
movement libras	**80.2 (1)**	72.5 (4)	76.1 (2)	73.8 (3)
new thyroid	92.6 (2)	92 (3)	**93 (1)**	91.1 (4)
pima	74.4 (2)	74.2 (3)	72.9 (4)	**74.5 (1)**
post operative	64.4 (4)	**70.1 (1.5)**	64.5 (3)	**70.1 (1.5)**
ring	**92.9 (1)**	91.5 (4)	92.8 (2)	92.7 (3)
saheart	69.2 (3)	**69.6 (1.5)**	68.4 (4)	**69.6 (1.5)**
sonar	76.9 (3)	74 (4)	**79.3 (1)**	77.8 (2)
spectfheart	79.4 (3)	79.4 (3)	79.4 (3)	**79.7 (1)**
splice	91.1 (4)	91.7 (3)	**94.2 (1)**	93.6 (2)
thyroid	93 (3)	92.8 (4)	**93.9 (1)**	93.3 (2)
titanic	76.8 (3)	72 (4)	**78.3 (1.5)**	**78.3 (1.5)**
vehicle	67.8 (3)	55.9 (4)	**73.1 (1)**	71.3 (2)
vowel	**82 (1)**	72.8 (4)	79.9 (3)	81.1 (2)
wdbc	94.5 (4)	94.7 (2.5)	94.7 (2.5)	**94.8 (1)**
wine	93.2 (2.5)	91.5 (4)	**93.8 (1)**	93.2 (2.5)
wisconsin	94.4 (4)	94.7 (2.5)	**95.4 (1)**	94.7 (2.5)
yeast	49.1 (3)	46.2 (4)	**56.8 (1.5)**	**56.8 (1.5)**
Mean	78.2375	77.545	80.9175	80.855

Table B.13: Results obtained by NSLV, NSLV-AR, NSLV-FR and SLAVE3 on 40 databases. The table shows the average number of rules.

Data	Rules			
	NSLV	NSLV-AR	NSLV-FR	SLAVE3
appendicitis	5.3 (4)	**3.7 (1)**	4.2 (2)	4.7 (3)
australian	12.4 (4)	**4.9 (1)**	10.7 (3)	6.9 (2)
automobile	18.2 (2.5)	**14.2 (1)**	18.2 (2.5)	19.2 (4)
balance	17.8 (4)	10.7 (2)	12.8 (3)	**8.5 (1)**
banana	6.3 (2)	**5.1 (1)**	22.3 (3)	24.8 (4)
breast	12.3 (4)	**4.6 (1)**	6.7 (3)	5 (2)
bupa	6.7 (2)	**4.3 (1)**	20.4 (4)	16.6 (3)
chess	5.3 (4)	**4.1 (1)**	4.3 (2)	4.6 (3)
cleveland	50.9 (4)	**8.5 (1)**	37 (3)	32.9 (2)
contraceptive	44.8 (2)	**3.2 (1)**	80.1 (4)	62.7 (3)
crx	7.7 (4)	**3.3 (1)**	5.1 (3)	4.8 (2)
dermatology	9.7 (2)	**8.9 (1)**	10.3 (4)	10.1 (3)
ecoli	14.6 (2)	**11.1 (1)**	27.1 (4)	24.5 (3)
flare	29.1 (4)	**4.4 (1)**	25.6 (3)	19.6 (2)
glass	15.1 (2)	**9.1 (1)**	16.6 (4)	16.4 (3)
haberman	4.8 (4)	2.9 (3)	2.8 (2)	**2.1 (1)**
hayes-roth	9 (2)	**8.4 (1)**	9.1 (3.5)	9.1 (3.5)
heart	11.2 (3)	**9.6 (1)**	12.5 (4)	10.2 (2)
housevotes	2.7 (2)	**2.2 (1)**	3.3 (3.5)	3.3 (3.5)
iris	3.5 (2)	**3 (1)**	4.6 (4)	4.3 (3)
led7digit	12.3 (2)	**11.4 (1)**	20.6 (3)	21.1 (4)
lymphography	10.4 (4)	9.3 (3)	9.1 (2)	**8.5 (1)**
monk-2	**2.5 (1)**	2.8 (2)	4.1 (4)	3.5 (3)
movement libras	49.9 (4)	**41.1 (1)**	48.1 (3)	47.5 (2)
new thyroid	6 (2.5)	6.1 (4)	6 (2.5)	**5.9 (1)**
pima	12 (2)	**4.6 (1)**	23 (4)	21.9 (3)
post operative	5.7 (4)	2.5 (2)	3.8 (3)	**2.3 (1)**
ring	29.3 (2)	**10.4 (1)**	76.4 (3)	77.2 (4)
saheart	14.7 (2)	**7.8 (1)**	24.9 (4)	20.5 (3)
sonar	10.8 (2)	**10.1 (1)**	12.2 (3)	12.9 (4)
spectfheart	2.1 (2)	**1 (1)**	2.2 (3.5)	2.2 (3.5)
splice	15.5 (3)	**5.2 (1)**	20.9 (4)	13 (2)
thyroid	3.2 (3)	**1.3 (1)**	3.3 (4)	2.7 (2)
titanic	3.7 (4)	2.5 (3)	**2.3 (1)**	2.4 (2)
vehicle	41.1 (2)	**16.6 (1)**	52.6 (4)	48.8 (3)
vowel	82.3 (3)	**60.4 (1)**	86.4 (4)	80.4 (2)
wdbc	8.1 (3)	**6.9 (1)**	8.3 (4)	7.8 (2)
wine	**6.5 (1.5)**	**6.5 (1.5)**	7.2 (3)	7.7 (4)
wisconsin	**8 (1.5)**	8.8 (3)	**8 (1.5)**	11 (4)
yeast	19.9 (2)	**9 (1)**	76.2 (4)	63.8 (3)
Mean	15.785	8.7625	20.7325	18.785

Table B.14: Results obtained by NSLV, NSLV-AR, NSLV-FR and SLAVE3 on 40 databases. The table shows the average number of conditions per rule.

Data	Conditions			
	NSLV	NSLV-AR	NSLV-FR	SLAVE3
appendicitis	2.2 (3.5)	1.9 (2)	2.2 (3.5)	**1.6 (1)**
australian	3.8 (4)	**2.4 (1)**	3.2 (3)	2.7 (2)
automobile	3.8 (2)	4 (3.5)	**3.7 (1)**	4 (3.5)
balance	2.4 (4)	**1.7 (1)**	2.2 (3)	2.1 (2)
banana	1.8 (2)	**1.7 (1)**	3.4 (3.5)	3.4 (3.5)
breast	2.9 (4)	**1.4 (1)**	2.7 (3)	2 (2)
bupa	3.4 (4)	**2.7 (1)**	3.3 (3)	3.1 (2)
chess	3.1 (4)	2.2 (3)	1.6 (2)	**1.4 (1)**
cleveland	4.7 (2.5)	**3.7 (1)**	5 (4)	4.7 (2.5)
contraceptive	4.4 (2)	**3.4 (1)**	5 (3)	5.1 (4)
crx	3.2 (4)	**1.3 (1)**	2.1 (2)	2.3 (3)
dermatology	3.3 (3)	3.7 (4)	**2.5 (1)**	2.7 (2)
ecoli	3.5 (2.5)	**2.5 (1)**	3.7 (4)	3.5 (2.5)
flare	3.2 (4)	**2.2 (1)**	2.9 (3)	2.6 (2)
glass	3.4 (4)	**2.6 (1)**	3.1 (3)	2.9 (2)
haberman	2 (4)	1.7 (3)	1 (2)	**0.2 (1)**
hayes-roth	2.1 (4)	**1.5 (1.5)**	1.7 (3)	**1.5 (1.5)**
heart	3.1 (4)	**2.7 (1.5)**	3 (3)	**2.7 (1.5)**
housevotes	0.9 (2)	**0.6 (1)**	1.3 (3.5)	1.3 (3.5)
iris	1.1 (2)	**0.8 (1)**	1.6 (4)	1.2 (3)
led7digit	4.2 (4)	3.1 (2.5)	3.1 (2.5)	**3 (1)**
lymphography	2.6 (4)	2.1 (2.5)	**2 (1)**	2.1 (2.5)
monk-2	**0.9 (1.5)**	**0.9 (1.5)**	1.2 (4)	1.1 (3)
movement libras	6.9 (2)	**6.3 (1)**	8.3 (4)	7.5 (3)
new thyroid	2.4 (4)	**1.6 (1)**	1.9 (3)	1.8 (2)
pima	3.3 (2.5)	**2.1 (1)**	3.7 (4)	3.3 (2.5)
post operative	2.3 (4)	**1.7 (1)**	2 (3)	1.8 (2)
ring	6.6 (4)	5.6 (2)	**5.1 (1)**	5.8 (3)
saheart	3.9 (3.5)	**2.5 (1)**	3.9 (3.5)	3.6 (2)
sonar	5 (3)	5.6 (4)	4.6 (2)	**4.2 (1)**
spectfheart	8.4 (4)	**0 (1)**	1 (3)	0.4 (2)
splice	3.9 (3)	**2.7 (1)**	4.8 (4)	3.2 (2)
thyroid	2.6 (4)	1.9 (2)	2.1 (3)	**0.9 (1)**
titanic	1.6 (3.5)	1.6 (3.5)	1 (2)	**0.8 (1)**
vehicle	5.4 (4)	**3.9 (1)**	4.2 (3)	4 (2)
vowel	6.2 (4)	**5.1 (1.5)**	5.6 (3)	**5.1 (1.5)**
wdbc	4 (4)	3.5 (3)	**2.7 (1.5)**	**2.7 (1.5)**
wine	3.4 (4)	**2.4 (1)**	2.8 (3)	2.7 (2)
wisconsin	2.1 (2)	**2 (1)**	2.9 (4)	2.6 (3)
yeast	**3.6 (1)**	3.8 (2)	5 (3)	5.3 (4)
Mean	3.44	2.5775	3.0775	2.8225

B.4.2 Results obtained by GCCL, C4.5, SGERD, FARC-HD, FURIA and SLAVE3 on 40 databases

Table B.15: Results obtained by GCCL, C4.5, SGERD, FARC-HD, FURIA and SLAVE3 on 40 databases. The table shows the accuracy on training set.

Data	Training					
	GCCL	C4.5	SGERD	FARC-HD	FURIA	SLAVE3
appendicitis	83.7 (6)	90.9 (3)	87.7 (5)	**93.9 (1)**	90.7 (4)	92.5 (2)
australian	75.4 (6)	**93 (1)**	85.5 (5)	90.6 (2)	89.3 (3)	89.2 (4)
automobile	62.6 (5)	93.7 (3)	59.6 (6)	**98.8 (1)**	91.2 (4)	98.1 (2)
balance	88.5 (4.5)	89.9 (3)	75.1 (6)	92.1 (2)	88.5 (4.5)	**96.6 (1)**
banana	75.2 (5)	**91.4 (1)**	60.7 (6)	86.3 (4)	89.5 (2)	89.2 (3)
breast	70.7 (5)	77.5 (3.5)	69.9 (6)	**91.3 (1)**	77.5 (3.5)	78.2 (2)
bupa	59.9 (5)	**86.4 (1)**	59.8 (6)	78.9 (3)	77.5 (4)	81.5 (2)
chess	0 (6)	99.6 (2)	61.7 (5)	97.5 (3)	**99.7 (1)**	94.3 (4)
cleveland	58.1 (5)	82.9 (3)	54.5 (6)	**87.7 (1)**	62.4 (4)	86.6 (2)
contraceptive	43.8 (6)	**73.2 (1)**	49.4 (5)	62.5 (3)	55.9 (4)	67 (2)
crx	73.3 (6)	90.8 (2)	86.3 (5)	**91.2 (1)**	89.3 (3)	88.9 (4)
dermatology	86.9 (5)	98.2 (4)	83.4 (6)	**99.9 (1)**	98.6 (3)	99.3 (2)
ecoli	67.8 (6)	91.7 (3)	76.9 (5)	92 (2)	90.2 (4)	**93.6 (1)**
flare	31 (6)	78.5 (3)	71.2 (5)	**79.8 (1)**	75.9 (4)	79.4 (2)
glass	68.5 (5)	**93.7 (1)**	62.8 (6)	81.5 (4)	86.3 (2)	83.8 (3)
haberman	73.8 (5)	75.8 (3)	74.4 (4)	**80.5 (1)**	76.5 (2)	73.6 (6)
hayes-roth	81.3 (5)	88.8 (3)	65.6 (6)	**91.5 (1)**	87.4 (4)	90.3 (2)
heart	89.4 (4)	91.7 (3)	75.5 (6)	**93.8 (1)**	88.8 (5)	92.5 (2)
housevotes	53.4 (6)	97.1 (4)	90.2 (5)	**98.3 (1.5)**	**98.3 (1.5)**	98 (3)
iris	95.7 (5)	98 (2)	95.1 (6)	**98.5 (1)**	97.8 (3)	97.7 (4)
led7digit	78 (2)	77.1 (3)	40.8 (6)	74.8 (5)	76.4 (4)	**79.1 (1)**
lymphography	73.2 (5)	92.3 (4)	69.6 (6)	**100 (1)**	95.1 (2)	94.9 (3)
monk-2	97.2 (5)	**100 (1.5)**	80.5 (6)	99.9 (3)	**100 (1.5)**	98.4 (4)
movement libras	19 (6)	94.3 (3)	51.7 (5)	95.5 (2)	91.8 (4)	**97.5 (1)**
new thyroid	87.4 (6)	98.5 (3)	88.5 (5)	98.8 (2)	**99.3 (1)**	97.6 (4)
pima	69.7 (6)	**83.1 (1)**	73.6 (5)	82.8 (2)	79.3 (4)	82.3 (3)
post operative	71.2 (5.5)	71.7 (4)	71.9 (3)	**90.5 (1)**	71.2 (5.5)	73.4 (2)
ring	90.5 (5)	98.6 (2)	72.2 (6)	95.2 (4)	**98.7 (1)**	96.4 (3)
saheart	68.1 (6)	78.8 (3)	70.7 (5)	**82.6 (1)**	75.5 (4)	82.5 (2)
sonar	75.9 (5)	97.7 (3)	73.7 (6)	**98.7 (1)**	98 (2)	97.2 (4)
spectfheart	79.4 (5)	**98.2 (1)**	79 (6)	91.5 (3)	94.3 (2)	79.9 (4)
splice	0 (6)	96.2 (3)	81.6 (5)	96.5 (2)	**99.4 (1)**	94.5 (4)
thyroid	92.6 (5)	**99.8 (1.5)**	92 (6)	94.2 (3)	**99.8 (1.5)**	93.4 (4)
titanic	78.2 (5)	78.4 (3)	71.9 (6)	**79 (1)**	78.6 (2)	78.3 (4)
vehicle	62 (5)	**88.9 (1)**	52.8 (6)	79.4 (4)	80.1 (3)	88.4 (2)
vowel	60.1 (5)	**97.1 (1)**	44.4 (6)	80.4 (4)	96.7 (2)	94.6 (3)
wdbc	91.8 (5)	99.1 (2)	90.9 (6)	98.6 (3)	**99.3 (1)**	98.4 (4)
wine	97.6 (5)	98.8 (4)	93.2 (6)	**99.8 (1.5)**	99.3 (3)	**99.8 (1.5)**
wisconsin	97.3 (5)	97.7 (4)	93.7 (6)	98.3 (2.5)	98.3 (2.5)	**98.7 (1)**
yeast	48.9 (5)	**81.5 (1)**	39.5 (6)	63.8 (3.5)	63.8 (3.5)	68.3 (2)
Mean	69.4275	90.265	71.9375	89.6725	87.655	89.0975

B.4. Chapter 6

Table B.16: Results obtained by GCCL, C4.5, SGERD, FARC-HD, FURIA and SLAVE3 on 40 databases. The table shows the accuracy on testing set.

Data	Test					
	GCCL	C4.5	SGERD	FARC-HD	FURIA	SLAVE3
appendicitis	83 (5)	83.2 (3)	83.1 (4)	87.1 (2)	**88.1 (1)**	82.8 (6)
australian	73.6 (6)	85.7 (3)	85.5 (4)	**86.2 (1)**	85.3 (5)	86 (2)
automobile	53.2 (5)	**80.9 (1)**	47.5 (6)	80.2 (2)	76.8 (4)	79.7 (3)
balance	81.9 (4)	76.7 (5)	75.9 (6)	86.5 (2)	82.9 (3)	**95.9 (1)**
banana	75.2 (5)	**89.1 (1)**	60.5 (6)	85.6 (4)	88.1 (3)	88.3 (2)
breast	70.8 (4)	**76.9 (1)**	70.4 (5)	75 (3)	75.4 (2)	70.3 (6)
bupa	58.4 (5)	**69.3 (1)**	57.3 (6)	63.7 (4)	68.6 (2)	67.9 (3)
chess	0 (6)	**99.4 (1.5)**	61.4 (5)	96.9 (3)	**99.4 (1.5)**	94.3 (4)
cleveland	54.4 (4.5)	54.4 (4.5)	46.8 (6)	56.5 (2)	56 (3)	**56.7 (1)**
contraceptive	43.3 (6)	53.8 (2)	47.9 (5)	52.3 (4)	**54.2 (1)**	53.1 (3)
crx	71.7 (6)	85.2 (5)	86.2 (2)	86.1 (3)	**86.6 (1)**	85.7 (4)
dermatology	82.1 (5)	94.3 (2)	81.2 (6)	91.8 (4)	**95.2 (1)**	93.5 (3)
ecoli	65.5 (6)	79.4 (4)	74.6 (5)	**82.1 (1)**	80.6 (2)	80 (3)
flare	31 (6)	74.2 (3)	70.7 (5)	73.8 (4)	74.8 (2)	**74.9 (1)**
glass	63.2 (5)	67.4 (3)	61 (6)	68.3 (2)	**70.4 (1)**	66.8 (4)
haberman	73.2 (4)	73.1 (5)	73.5 (3)	72.5 (6)	**75.4 (1)**	73.8 (2)
hayes-roth	74.3 (5)	80 (2.5)	44.9 (6)	79.3 (4)	80 (2.5)	**81.8 (1)**
heart	78.8 (3)	78.5 (4)	77 (6)	**85.1 (1)**	82.5 (2)	77.7 (5)
housevotes	52.9 (6)	**97 (1)**	87.4 (5)	94.5 (4)	96.6 (2)	95.7 (3)
iris	95.3 (3.5)	**96 (1.5)**	95.3 (3.5)	**96 (1.5)**	94 (6)	94.6 (5)
led7digit	71.2 (2)	71 (3)	40 (6)	66.8 (5)	69.4 (4)	**71.6 (1)**
lymphography	68.5 (5)	74.2 (4)	65.4 (6)	76 (3)	**81.5 (1)**	79.2 (2)
monk-2	97.2 (5)	**100 (1.5)**	80.6 (6)	99.5 (3)	**100 (1.5)**	97.5 (4)
movement libras	15.2 (6)	69.4 (3)	42.5 (5)	73.3 (2)	66.1 (4)	**73.8 (1)**
new thyroid	86.1 (6)	92.5 (3)	87 (5)	**94.3 (1)**	93.5 (2)	91.1 (4)
pima	68.2 (6)	73.9 (4)	73 (5)	75.7 (2)	**75.9 (1)**	74.5 (3)
post operative	**71.2 (1)**	68.8 (4.5)	69 (3)	57 (6)	68.8 (4.5)	70.1 (2)
ring	91 (4)	90.2 (5)	71.4 (6)	**94 (1)**	93.8 (2)	92.7 (3)
saheart	65.5 (6)	67 (5)	68.5 (4)	69.6 (2.5)	**70.8 (1)**	69.6 (2.5)
sonar	61 (6)	70.5 (4)	66.6 (5)	**80.6 (1)**	76.9 (3)	77.8 (2)
spectfheart	79.4 (3.5)	76.4 (6)	78.2 (5)	**83.1 (1)**	79.4 (3.5)	79.7 (2)
splice	0 (6)	94.1 (2.5)	81.8 (5)	94.1 (2.5)	**94.8 (1)**	93.6 (4)
thyroid	92.5 (5)	99.5 (2)	91.9 (6)	94.1 (3)	**99.6 (1)**	93.3 (4)
titanic	78.3 (3.5)	77.3 (5)	71.9 (6)	**78.8 (1)**	78.5 (2)	78.3 (3.5)
vehicle	57.2 (5)	**74.1 (1)**	51.8 (6)	67.9 (4)	71.6 (2)	71.3 (3)
vowel	54.6 (5)	**81.5 (1)**	38.6 (6)	71.6 (4)	80.1 (3)	81.1 (2)
wdbc	91.3 (5)	95.2 (3)	90.6 (6)	**95.9 (1.5)**	**95.9 (1.5)**	94.8 (4)
wine	91 (6)	94.9 (2)	92 (5)	94.2 (3)	**97.1 (1)**	93.2 (4)
wisconsin	**97.2 (1)**	94.7 (4.5)	93.5 (6)	96.6 (2)	95.7 (3)	94.7 (4.5)
yeast	47.4 (5)	55.5 (4)	38.8 (6)	**59.5 (1)**	59.4 (2)	56.8 (3)
Mean	66.645	80.38	69.53	80.5525	81.4925	80.855

Table B.17: Results obtained by GCCL, C4.5, SGERD, FARC-HD, FURIA and SLAVE3 on 40 databases. The table shows the average number of rules.

Data	Rules					
	GCCL	C4.5	SGERD	FARC-HD	FURIA	SLAVE3
appendicitis	21 (6)	3.2 (2)	**2.5 (1)**	6.6 (5)	3.9 (3)	4.7 (4)
australian	45.7 (6)	9.8 (4)	**2 (1)**	25.4 (5)	8.4 (3)	6.9 (2)
automobile	36.8 (5)	15.7 (2)	**7.1 (1)**	37.5 (6)	16 (3)	19.2 (4)
balance	87.1 (6)	35.5 (4)	**3.8 (1)**	63.3 (5)	22.6 (3)	8.5 (2)
banana	19 (4)	42.4 (6)	**2.4 (1)**	12.3 (2)	18.2 (3)	24.8 (5)
breast	**1 (1)**	14.1 (5)	2 (2)	46.9 (6)	4.7 (3)	5 (4)
bupa	34.4 (6)	9 (2)	**2.6 (1)**	10.3 (4)	9.1 (3)	16.6 (5)
chess	**0 (1)**	31 (5)	2 (2)	32.8 (6)	28.3 (4)	4.6 (3)
cleveland	56.5 (5)	9.7 (3)	**6.7 (1)**	61.4 (6)	8.8 (2)	32.9 (4)
contraceptive	76.3 (6)	29.6 (3)	**4.5 (1)**	73.9 (5)	8.7 (2)	62.7 (4)
crx	28 (6)	12.8 (4)	**2 (1)**	24.5 (5)	7.4 (3)	4.8 (2)
dermatology	54.8 (6)	9.2 (2)	**8.2 (1)**	28.2 (5)	10.9 (4)	10.1 (3)
ecoli	62.2 (6)	11.6 (2)	**8.9 (1)**	34 (5)	17 (3)	24.5 (4)
flare	**1 (1)**	25.3 (5)	6.1 (2)	46.1 (6)	9.4 (3)	19.6 (4)
glass	49 (6)	10.3 (2)	**6.1 (1)**	23.8 (5)	12.1 (3)	16.4 (4)
haberman	10.1 (5)	4.3 (4)	3.1 (2)	10.7 (6)	3.7 (3)	**2.1 (1)**
hayes-roth	28.3 (6)	10 (4)	**5.3 (1)**	18.5 (5)	9 (2)	9.1 (3)
heart	69 (6)	10.1 (3)	**2.7 (1)**	27.3 (5)	8.6 (2)	10.2 (4)
housevotes	**1 (1)**	5.3 (5)	2 (2)	9.6 (6)	5.1 (4)	3.3 (3)
iris	12.8 (6)	5 (5)	**3.9 (1)**	4.4 (3)	4.5 (4)	4.3 (2)
led7digit	64 (6)	16.1 (3)	**7.1 (1)**	26.3 (5)	13.4 (2)	21.1 (4)
lymphography	8.7 (3)	11.4 (4)	**4.2 (1)**	22.4 (6)	14 (5)	8.5 (2)
monk-2	8.9 (5)	6 (4)	**2.1 (1)**	13.4 (6)	5 (3)	3.5 (2)
movement libras	25.7 (2)	27.3 (3)	**20.9 (1)**	82.7 (6)	32.6 (4)	47.5 (5)
new thyroid	28 (6)	6.6 (3)	**3.3 (1)**	9.5 (5)	6.8 (4)	5.9 (2)
pima	65.1 (6)	9.2 (3)	**2.9 (1)**	24.7 (5)	9.1 (2)	21.9 (4)
post operative	**1 (1)**	4.6 (5)	2.1 (2)	23.5 (6)	2.5 (4)	2.3 (3)
ring	85.7 (4)	103.3 (6)	**5.2 (1)**	23.8 (2)	87.3 (5)	77.2 (3)
saheart	68.8 (6)	6.9 (3)	**2.7 (1)**	29.3 (5)	6.8 (2)	20.5 (4)
sonar	51.1 (6)	8.9 (2)	**3.4 (1)**	18.2 (5)	10.6 (3)	12.9 (4)
spectfheart	7.4 (3)	11.7 (4)	2.4 (2)	14.1 (6)	12.8 (5)	**2.2 (1)**
splice	**0 (1)**	135.7 (5)	4.3 (2)	79.8 (4)	143.6 (6)	13 (3)
thyroid	23.3 (6)	10.4 (4)	**2 (1)**	4.8 (3)	14.9 (5)	2.7 (2)
titanic	10 (6)	5 (4)	**2 (1)**	4.5 (3)	5.7 (5)	2.4 (2)
vehicle	80 (6)	19.7 (2)	**6.4 (1)**	45.4 (4)	24.4 (3)	48.8 (5)
vowel	82.4 (6)	52.9 (2)	**18.3 (1)**	73.4 (4)	60.6 (3)	80.4 (5)
wdbc	60 (6)	9.5 (3)	**3.4 (1)**	10.9 (4)	11.6 (5)	7.8 (2)
wine	61.7 (6)	5 (2)	**4 (1)**	9.1 (5)	6.4 (3)	7.7 (4)
wisconsin	30.2 (6)	10 (2)	**2.2 (1)**	13.6 (4)	14.4 (5)	11 (3)
yeast	75.1 (6)	35.4 (4)	**12.5 (1)**	34.9 (3)	23.4 (2)	63.8 (5)
Mean	38.2775	19.9875	4.8825	29.045	18.0575	18.785

B.4.3 Results obtained by NSLV-AR and SLAVE3* on 40 databases

Table B.18: Results obtained by NSLV-AR and SLAVE3* on 40 databases. The table shows the accuracy on training and testing sets.

	Training		Test	
Dataset	NSLV-AR	SLAVE3*	NSLV-AR	SLAVE3*
appendicitis	91.9 (2)	**92 (1)**	**87.1 (1)**	86.2 (2)
australian	87.4 (2)	**88.5 (1)**	85.3 (2)	**86 (1)**
automobile	**97.3 (1)**	95.6 (2)	**76 (1)**	73.1 (2)
balance	81.1 (2)	**82.8 (1)**	75.3 (2)	**76.7 (1)**
banana	75.2 (2)	**76.8 (1)**	74.6 (2)	**76.8 (1)**
breast	75.9 (2)	**78.1 (1)**	71.1 (2)	**71.2 (1)**
bupa	66.5 (2)	**66.6 (1)**	55.2 (2)	**56.7 (1)**
chess	**94.4 (1)**	94.1 (2)	**94 (1.5)**	**94 (1.5)**
cleveland	62.8 (2)	**69 (1)**	**56.1 (1)**	53.1 (2)
contraceptive	35.2 (2)	**36 (1)**	34.3 (2)	**35.4 (1)**
crx	**86.2 (1)**	84.2 (2)	**85.3 (1)**	82.4 (2)
dermatology	**99.1 (1.5)**	**99.1 (1.5)**	**94.4 (1)**	93 (2)
ecoli	83.6 (2)	**84.7 (1)**	77 (2)	**77.9 (1)**
flare	64.5 (2)	**67.9 (1)**	63.8 (2)	**67 (1)**
glass	74.9 (2)	**75 (1)**	62.6 (2)	**64 (1)**
haberman	**76.7 (1)**	75.9 (2)	74.1 (2)	**74.4 (1)**
hayes-roth	**90.6 (1)**	89.7 (2)	**80.6 (1)**	79.3 (2)
heart	90.8 (2)	**90.9 (1)**	79.6 (2)	**82.5 (1)**
housevotes	97 (2)	**97.6 (1)**	**97 (1)**	96.4 (2)
iris	**96.5 (1.5)**	**96.5 (1.5)**	**95.3 (1.5)**	**95.3 (1.5)**
led7digit	70.5 (2)	**72.4 (1)**	66.3 (2)	**67.9 (1)**
lymphography	95 (2)	**95.4 (1)**	**83.3 (1)**	80.4 (2)
monk-2	**98 (1)**	97.4 (2)	**97.9 (1)**	97.7 (2)
movement libras	**96.7 (1)**	96.2 (2)	**72.5 (1)**	71.1 (2)
new thyroid	**95.1 (1)**	91.7 (2)	**92 (1)**	87.4 (2)
pima	76.6 (2)	**76.9 (1)**	**74.2 (1)**	72.9 (2)
post operative	73.4 (2)	**75.7 (1)**	**70.1 (1)**	66.6 (2)
ring	92.1 (2)	**95.1 (1)**	91.5 (2)	**92.6 (1)**
saheart	76.5 (2)	**77.2 (1)**	**69.6 (1)**	67.1 (2)
sonar	**98.6 (1)**	97.7 (2)	74 (2)	**80.7 (1)**
spectfheart	**79.4 (1.5)**	**79.4 (1.5)**	**79.4 (1.5)**	**79.4 (1.5)**
splice	92.1 (2)	**92.8 (1)**	91.7 (2)	**92.1 (1)**
thyroid	92.8 (2)	**93.4 (1)**	92.8 (2)	**93.4 (1)**
titanic	72.2 (2)	**74.1 (1)**	72 (2)	**74.4 (1)**
vehicle	64.1 (2)	**76.6 (1)**	55.9 (2)	**64.3 (1)**
vowel	86.2 (2)	**90 (1)**	72.8 (2)	**75.6 (1)**
wdbc	98 (2)	**98.4 (1)**	94.7 (2)	**95.6 (1)**
wine	**99.6 (1)**	99 (2)	91.5 (2)	**93.1 (1)**
wisconsin	**98.4 (1)**	98.1 (2)	**94.7 (1)**	93.8 (2)
yeast	49.1 (2)	**51 (1)**	46.2 (2)	**50.7 (1)**
Mean	83.3	84.2375	77.545	77.955

Table B.19: Results obtained by NSLV-AR and SLAVE3* on 40 databases. The table shows the average number of rules and the time employed to get the model measured in seconds.

	Rules		Time	
Dataset	NSLV-AR	SLAVE3*	NSLV-AR	SLAVE3*
appendicitis	**3.7 (1)**	3.8 (2)	1.1 (2)	**0.9 (1)**
australian	**4.9 (1)**	6.8 (2)	**7 (1)**	7.6 (2)
automobile	**14.2 (1)**	14.3 (2)	9.9 (2)	**9 (1)**
balance	**10.7 (1)**	11.4 (2)	4.1 (2)	**3.6 (1)**
banana	**5.1 (1)**	6.4 (2)	**14.8 (1)**	18.4 (2)
breast	**4.6 (1)**	5.6 (2)	1.9 (2)	**1.7 (1)**
bupa	4.3 (2)	**4.2 (1)**	2.1 (2)	**1.9 (1)**
chess	**4.1 (1)**	5.6 (2)	**17.6 (1)**	31.9 (2)
cleveland	**8.5 (1)**	12.8 (2)	**6.4 (1)**	8.4 (2)
contraceptive	**3.2 (1)**	3.6 (2)	**4.2 (1)**	6 (2)
crx	**3.3 (1)**	4 (2)	**3.2 (1)**	4.4 (2)
dermatology	**8.9 (1)**	9.3 (2)	8.3 (2)	**7.1 (1)**
ecoli	**11.1 (1)**	12.6 (2)	6.2 (2)	**6 (1)**
flare	**4.4 (1)**	6.8 (2)	**3.6 (1)**	6.1 (2)
glass	**9.1 (1)**	9.3 (2)	5.5 (2)	**5.3 (1)**
haberman	**2.9 (1.5)**	**2.9 (1.5)**	1.1 (2)	**1 (1)**
hayes-roth	8.4 (2)	**7.6 (1)**	1.5 (2)	**1.2 (1)**
heart	9.6 (2)	**9.1 (1)**	4.2 (2)	**3.9 (1)**
housevotes	**2.2 (1)**	3.3 (2)	**0.6 (1)**	0.8 (2)
iris	**3 (1)**	3.2 (2)	**0.6 (1.5)**	**0.6 (1.5)**
led7digit	**11.4 (1)**	12.1 (2)	5.9 (2)	**5.7 (1)**
lymphography	**9.3 (1.5)**	**9.3 (1.5)**	2.6 (2)	**2 (1)**
monk-2	2.8 (2)	**2.6 (1)**	**0.9 (1.5)**	**0.9 (1.5)**
movement libras	**41.1 (1.5)**	**41.1 (1.5)**	**152.7 (1)**	162.7 (2)
new thyroid	6.1 (2)	**5.1 (1)**	1.6 (2)	**1.3 (1)**
pima	**4.6 (1)**	5.8 (2)	**5.8 (1)**	7.1 (2)
post operative	**2.5 (1)**	3.2 (2)	**0.6 (1.5)**	**0.6 (1.5)**
ring	**10.4 (1)**	34.9 (2)	**253.4 (1)**	1034.6 (2)
saheart	**7.8 (1)**	9.9 (2)	**5.7 (1)**	7.8 (2)
sonar	**10.1 (1)**	10.7 (2)	**17.4 (1)**	18.7 (2)
spectfheart	**1 (1.5)**	**1 (1.5)**	**0.7 (1)**	1.1 (2)
splice	**5.2 (1)**	6.2 (2)	**25 (1)**	35.3 (2)
thyroid	**1.3 (1)**	1.9 (2)	**13.6 (1)**	32.6 (2)
titanic	2.5 (2)	**2.4 (1)**	**2.8 (1)**	3.7 (2)
vehicle	**16.6 (1)**	28.2 (2)	**38.9 (1)**	62.4 (2)
vowel	**60.4 (1)**	69.1 (2)	**113.9 (1)**	119.8 (2)
wdbc	**6.9 (1)**	8.4 (2)	**15.5 (1)**	16.8 (2)
wine	6.5 (2)	**6.3 (1)**	2.4 (2)	**1.9 (1)**
wisconsin	8.8 (2)	**8.4 (1)**	8.4 (2)	**7.4 (1)**
yeast	**9 (1)**	9.1 (2)	**15.2 (1)**	16.4 (2)
Mean	8.7625	10.4575	19.6725	41.615

Bibliography

[1] Akbarzadeh, V., Sadeghian, A., dos Santos, M. V., "Derivation of relational fuzzy classification rules using evolutionary computation", *IEEE International Conference on Fuzzy Systems*, 1689–1693 (2008). 74

[2] Alcalá-Fdez, J., Alcalá, R., Herrera, F., "A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning", *IEEE Transactions on Fuzzy Systems*, **19 (5)**, 857–872 (2011). 140

[3] Alcalá-Fdez, J., Fernández, A., Luego, J. Derrac, J. and García S., "Keel data-mining software tool: Data set repository, integration and algorithms and experimental analysis framework", *Multiple-Valued Logic and Soft Computing*, **17 (2-3)**, 255–287 (2011). 27, 58, 62, 155

[4] Alcalá, R., Casillas, J., Cordón, O., Herrera, F., Zwir, I., "Techniques for learning and tuning fuzzy rule-based systems for linguistic modeling and their application", *Knowledge Engineering Systems, Techniques and Applications*, **3**, 889–941 (1999). 22

[5] Alfred, R., "Feature transformation: A genetic-based feature construction method for data summarization", *Computational Intelligence*, **27 (3)**, 315–335 (2011). 73

[6] Ali, S. H., "Miner for OACCR: Case of medical data analysis in knowledge discovery", *2012 6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT 2012)*, 6482043, 962–975 (2012). 73

[7] Assadi, A., Zade, S. H., "UGA: A new genetic algorithm-based classification method for uncertain data", *Middle-East Journal of Scientific Research*, **20 (10)**, 1207–1212 (2014). 111

[8] Asuncion, A., Newman, D. J., "UCI Machine Learning Repository", Irvine, CA: University of California, School of Information and Computer Science (2007). Available: http://archive.ics.uci.edu/ml/datasets/Iris. 4, 12

[9] Azam, A., Khan, M. H., "Reduced rule fuzzy logic controller for performance improvement of process control", *IEEE Conference on Information and Communication Technologies, ICT 2013*, 894–898 (2013). 112

[10] Back, T., "Evolutionary Algorithms in Theory and Practice", Oxford Univ. Press (1996). 27

[11] Bloedorn, E., Michalski, R. S., "Data Driven constructive induction: A methodology and applications", *Feature extraction. Constructive and Selection: a data mining perspective*, Kluwer, 51–68 (1998). 69

[12] Booker, L. B., Goldberg, D. E., Holland, J. H., "Classifier Systems and Genetic Algorithms", *Artificial Intelligence*, **40**, 235–282 (1989). 24

[13] Breiman, L., "Random Forests", *Machine Learning*, **45**, 5–32 (2001). 105

[14] Bresso, E., Grisoni, R., Devignes, M. -D., Napoli, A., Small-Tabone, M., "ILP Characterization of 3D Protein-Binding Sites and FCA-Based Interpretation", *Communications in Computer and Information Science*, 415, 84–100 (2013) . 72

[15] Caises, Y., Leyva, E., González, A., Pérez, R., "A genetic learning of fuzzy relational rules", *2010 IEEE World Congress on Computational Intelligence, WCCI 2010*, 1–8 (2010). xi, 74, 75

[16] Carse, B., Fogarty, T. C., Munro, A., "Evolving Fuzzy Rule Based Controllers Using Genetic Algorithms", *Fuzzy Sets and Systems*, **80**, 273–293 (1996). 25

[17] Casillas, J., Cordn, O., Herrera, F., "COR: A methodology to improve ad hoc data-driven linguistic rule learning methods by inducing cooperation among rules", *IEEE Transactions on Systems, Man and Cybernetics*, **32 (4)**, 526–537 (2002). 21

[18] Castillo L., González A., Pérez R., "Including a simplicity criterion in the selection of the best rule in a genetic fuzzy learning algorithm", *Fuzzy Sets and Systems*, **120 (2)**, 309–321 (2001). 46, 53, 54

[19] Cohen, W., "Fast effective rule induction", *Proceedings of the 12th international conference on machine learning, ICML, Morgan Kaufmann*, 115–123 (1995). 140

[20] Cordón, O., Del Jesús, M. J., Herrera, F., Lozano, M., "Genetic learning of fuzzy rule-based classification systems cooperating with fuzzy reasoning methods", *International Journal of Intelligent Systems*, **13 (10-11)**, 1025–1053 (1998). 27

[21] Cordón, O., Del Jesús, M. J., Herrera, F., Lozano, M., "MOGUL: A Methodology to Obtain Genetic Fuzzy Rule-Based Systems under the Iterative Rule Learning Approach", *Tech. Rep. DECSAI-980101, Dept. Computer Sciences and Artificial Intelligence, University of Granada* (1998). 27

[22] Cordón, O., Herrera, F., Gomide, F., Hoffmann, F., Magdalena, L., "Ten years of genetic fuzzy systems: current framework and new trends", *In IFSA World Congress and 20th NAFIPS International Conference*, **3**, 1241–1246 (2001). xi, 23, 24, 25

[23] Cordón, O., Herrera, F., Hoffmann, F., Magdalena, L., "Genetic fuzzy systems", *Singapore: World Scientific Publishing Company* (2001). 19, 21, 24

[24] Cordón, O., Herrera, F., Hoffmann, F., Magdalena, L., "Genetic Fuzzy Systems", *Singapore: World Scientific Publishing Company* (2001). 5, 12

[25] Dor, O., Reich, Y., "Strengthening learning algorithms by feature discovery", *Information Sciences*, **189**, 176–190 (2012). 79

[26] Dubois, D., Prade, H., "What are fuzzy rules and how to use them", *Fuzzy Sets and Systems*, **84(2)**, 169–185 (1996). 4, 12

[27] Fazzolari, M., Alcalá, R., Nojima, Y., Ishibuchi, H., Herrera, F., "Improving a fuzzy association rule-based classification model by granularity learning based on heuristic measures over multiple granularities", *Proceedings of the 2013 IEEE International Workshop on Genetic and Evolutionary Fuzzy Systems, GEFS 2013 - 2013 IEEE Symposium Series on Computational Intelligence, SSCI*, 44–51 (2013). 112

[28] França, M. V. M., Zaverucha, G., D'Avila Garcez, A. S., "Fast relational learning using bottom clause propositionalization with artificial neural networks", *Machine Learning*, **94 (1)**, 81–104 (2014). 72

[29] Friedman, J., Hastie, T., Tibshirani, R., "Additive logistic regression: a statistical view of boosting", *Ann Stat*, **38 (2)**, 337–374 (2000). 28

[30] Friedman, M., "A comparison of alternative tests of significance for the problem of m rankings", *Annals of Mathematical Statistics*, **11**, 86–92 (1940). 155

[31] Friedman, M., "The use of ranks to avoid the assumption of normality implicit in the analysis of variance", *Journal of the American Statistical Association*, **32**, 674–701 (1937). 155

[32] Gacto, M. J., Alcalá, R., Herrera, F., "A double axis classification of interpretability measures for Linguistic Fuzzy Rule-Based Systems", *Lecture Notes in Computer Science*, 99–106 (2011). 111, 112

[33] Galende-Hernández, M., Sainz-Palmero, G. I., Fuente-Aparicio, M. J., "Complexity reduction and interpretability improvement for fuzzy rule systems based on simple interpretability measures and indices by bi-objective evolutionary rule selection", *Soft Computing*, **16(3)**, 451–470 (2012). 112

[34] García, D., González, A., Pérez, R., "A feature construction approach for genetic iterative rule learning algorithm", *Journal of Computer and System Sciences*, **80(1)**, 101–117 (2014). 89

[35] García, D., González A., Pérez R., "An empirical study about the behavior of a genetic learning algorithm on searching spaces pruned by a completeness condition", *Proceedings of the 2013 IEEE International Workshop on Genetic and Evolutionary Fuzzy Systems, GEFS 2013 - 2013 IEEE Symposium Series on Computational Intelligence, SSCI*, 8–15 (2013). 118, 120

[36] García, D., González, A., Pérez, R., "A new iterative model to simplify the knowledge extracted on a fuzzy rule-based learning algorithm", *IEEE International Conference on Fuzzy Systems* (2013). 113

[37] García, D., González, A., Pérez, R., "Overview of the SLAVE learning algorithm: A review of its evolution and prospects", *International Journal of Computational Intelligence Systems*, **7 (6)**, 1194–1221 (2014). 5, 13, 25, 75

[38] Gaweda, A. E., Zurada, J., M., "Data-driven linguistic modeling using relational fuzzy rules", *IEEE Transactions on Fuzzy Systems*, **11 (1)**, 121–134 (2003). 74

[39] Ghannadpour, S. F., Noori, S., Tavakkoli-Moghaddam, R., Ghoseiri, K., "A multi-objective dynamic vehicle routing problem with fuzzy time windows: Model, solution and application", *Applied Soft Computing Journal*, **14**, 504–527 (2014). 111

[40] Giannoglou, V. G., Stavrakoudis, D. G., Theocharis, J. B., Petridis, V., "Genetic fuzzy rule-based classification systems for tissue characterization of intravascular ultrasound images", *IEEE International Conference on Fuzzy Systems*, 1–8 (2012). 111

[41] Goldberg, D.E., Genetic Algorithms in Search, Optimization, and Machine Lear- ning. Addison-Wesley, New York, (1989). 22

[42] González, A., Herrera, F., "Multi-stage genetic fuzzy system based on the iterative rule learning approach", *Mathware and Soft Computing*, **4 (3)**, 233–249 (1997). 24

[43] González, A., Pérez, R., "A learning system of fuzzy control rules", in: Herrera, F., Verdegay, J. L. (Eds.), *Genetic Algorithms and Soft Computing, Physica-Verlag, Wurzburg*, 202–225 (1996). 33, 35, 52, 53

[44] González, A., Pérez, R., "A two level genetic fuzzy learning algorithm for solving complex problem", *Proc. IFSA'97, Prague*, 192–197 (1997). 28, 52

[45] González, A., Pérez, R., "Completeness and consistency conditions for learning fuzzy rules ", *Fuzzy Set and Systems*, **96**, 37–51 (1998). 5, 13, 25, 35, 38, 53

[46] González, A., Pérez, R., "Improving the genetic algorithm of SLAVE", *Mathware & Soft Computing*, **16**, 59–70 (2009). 5, 13, 57

[47] González, A., Pérez, R., "Selection of relevant features in a fuzzy genetic learning algorithm", *IEEE Trans. on Systems, Man and Cybernetics-Part. B Cybernetics*, **31 (3)**, 417–425 (2001). 46, 48, 51

[48] González, A., Pérez, R., "SLAVE: A genetic learning system based on an iterative approach", *IEEE Trans. on Fuzzy Systems*, **7 (2)**, 176–191 (1999). 39, 46, 47

[49] González, A., Pérez, R., Caises,Y., and Leyva E., "An Efficient Inductive Genetic Learning Algorithm for Fuzzy Relational Rules", *International Journal of Computational Intelligence Systems*, 212–230 (2012). 74

[50] González, A., Pérez, R., Valenzuela, A., "Diagnosis of myocardial infarction through fuzzy learning techniques", *Proc. IFSA'95, Sao Paulo*, **1**, 273–276 (1995). 52

[51] González, A., Pérez, R., Verdegay, J. L., "Learning the structure of a fuzzy rule: A genetic approach", *Fuzzy Systems and A.I.*, **3 (1)**, 57–70 (1994). 33, 36, 39

[52] Haboudane, D., Miller, J. R., Tremblay, N., Vigneault, P., "Indices-based approach for crop chlorophyll content retrieval from hyperspectral data", *Proceedings of the 2007 International Geoscience and Remote Sensing Symposium* (IGARSS 2007), 3297–3300 (2007). 103

[53] Haralick, R. M., Shanmugam, K., Dinstein, I., "Textural Features for Image Classification", *IEEE Transactions on Systems, Man and Cybernetics*, **6**, 610–621 (1973). 105

[54] Haykin, S., "Neural Networks", Macmillan (1994). 22

[55] Holland, J. H., "Adaptation in natural and artificial systems", *Ann Arbor: University of Michigan Press* (1975). 22

[56] Holland J. H., "Escaping brittleness: the possibilities of general purpose learning algorithms applied", *Machine Learning: An artificial intelligence approach*, Morgan Kaufmann (1986). 24

[57] Holm, S., "A simple sequentially rejective multiple test procedure", *Scandinavian Journal of Statistics*, **6**, 65–70 (1979). 155

[58] Horne, J. H., "A tasseled cap transformation for IKONOS images", *Proceedings of the 2003 ASPRS Annual Conference* (ASPRS), Anchorage (Alaska) (2003). 105

[59] Hu, Y-J., Kliber, D., "Generation of attributes for learning algorithms", *Proceedings of the National Conference on Artificial Intelligence*, **1**, 806–811, 1996. 70

[60] Hühn, J., Hüllermeier, E., "FURIA: an algorithm for unordered fuzzy rule induction", *Data Mining and Knowledge Discovery*, **19 (3)**, 293–319 (2009). 140

[61] Igel, C., Hüsken, M., "Empirical evaluation of the improved Rprop learning algorithms", *Neurocomputing*, **50**, 105–123 (2003). 22

[62] Ishibuchi, H., Nakashima, T., Murata, T., "Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems", *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **29 (5)**, 601–618 (1999). 28

[63] Jang, J. S., "ANFIS: Adaptive-Network-Based Fuzzy Inference System", *IEEE Transactions on Systems, Man and Cybernetics*, **23 (3)**, 665–685 (1993). 22, 33

[64] Joshi, S., Ramakrishnan, G., Srinivasan, A., "Feature construction using theory-guided sampling and randomised search", *Lecture Notes in Computer Sciences*, 5194 LNAI, 140–157 (2008). 71

[65] Kamath, U., Compton, J., Islamaj-Doğan, R., De Jong, K., A., Shehu, A., "An evolutionary algorithm approach for feature generation from sequence data and its application to DNA splice site prediction", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **9 (5)**, 1387–1398 (2012). 73

[66] Koza, J., "Genetic Programming", *MIT Press* (1992). 73

[67] Krawiec, K., "Genetic programming-based construction of features for machine learning and knowledge discovery tasks", *Genetic Programming and Evolvable Machines*, **3 (4)**, 329–343 (2002). 72

[68] Kröse, B. J. A., Van der Smagt, P. D., "An introduction to neural networks", University of Amsterdam (1993). 22

[69] Kullback, S., "Information Theory and Statistics", *Gloucester, Mass* (1978). 83

[70] Kuželka, O., Železný, F., "Block-wise construction of tree-like relational features with monotone reducibility and redundancy", *Machine Learning*, **83 (2)**, 163–192 (2011). 72

[71] Lavrac, N., Dzeroski, S., Grobelnik, M., "Learning non-recursive definitions of relations with LINUS", *In EWSL-91: Proceedings of the European working session on learning on Machine learning*, Springer-Verlag New York, Inc., 265–281 (1991). 71

[72] Mamdani, E. H., "Applications of fuzzy algorithm for control a simple dynamic plant", *Proc. of the IEEE*, **121**, 1585–1588 (1974). 19

[73] Mamdani, E. H., Assilian, S., "An experiment in linguistic synthesis with fuzzy logic controller". *International Journal of Man-Machine Studies*, **1**, 1–13 (1975). 19, 27

[74] Mansoori, E. G., Zolghadri, M. J., Katebi, S. D., "SGERD: A steady-state genetic algorithm for extracting fuzzy classification rules from data", *IEEE Transactions on Fuzzy Systems*, **16(4)**, 1061–1071 (2008). 29

[75] Markovitch, S., Rosenstein, D., "Feature generation using general constructor functions", *Machine Learning*, **49 (1)**, 59–98 (2002). 71

[76] Martínez-Estudillo, F. J., Hervás-Martínez, C., Gutiérrez, P. A., Martínez-Estudillo, A. C., "Evolutionary product-unit neural networks classifiers", *Neurocomputing*, **72 (1)**, 548–561, (2008). 22

[77] Matheus, C., Rendell, L. A., "Constructive induction on decision trees", *In Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, 645–650 (1989). 70

[78] Michalski, R. S., "A theory and methodology of inductive learning", *Springer Berlin Heidelberg*, 83–134 (1983). 4, 11, 38

[79] Michalski, R. S., Mozetic, I., Hong, J., Lavrac, N., "The multi-purpose incremental learning system AQ15 and its testing application to three medical domains", *Proceedings of AAA-86 Fifth National Conference on Artificial Intelligence*, Morgan Kaufmann, 1041–1045 (1986). 41

[80] Michalski, R. S., "Understanding the nature of learning: Issues and research directions", *Machine learning: An artificial intelligence approach*, **2**, 3–25 (1986). 3, 11

[81] Mitchell, T., "Machine Learning", *Ed. MacGraw-Hill* (1997). 5, 13, 26, 34

[82] Muggleton, S., De Raedt, L., "Inductive Logic Programming: Theory and methods", *The Journal of Logic Programming*, 19-20 (SUPPL. 1), 629–679 (1994). 71

[83] Muharram, M., Smith, G. D., "Evolutionary constructive induction", *IEEE Transactions on Knowledge and Data Engineering*, **17 (11)**, 1518–1528 (2005). 79

[84] Müller, B., Reinhardt, J., "Neural networks. An introduction", Springer-Verlag (1990). 22

[85] Mylonas, S. K., Stavrakoudis, D. G., Theocharis, J. B., "A GA-based sequential fuzzy segmentation approach for classification of remote sensing images", *IEEE International Conference on Fuzzy Systems*, 1–8 (2012). 111

[86] Nauck, D., Klawonn, F., Kruse, R., "Foundations of neuro-fuzzy systems", John Wiley & Sons (1997). 22

[87] Nojima, Y., Ishibuchi, H., "Multiobjective genetic fuzzy rule selection with fuzzy relational rules", *Proceedings of the 2013 IEEE International Workshop on Genetic and Evolutionary Fuzzy Systems (GEFS)*, 60–67 (2013). 74

[88] Nozaki, K., Ishibuchi, H., Tanaka, H., "A simple but powerful heuristic method for generating fuzzy rules from numerical data", *Fuzzy Sets and Systems*, **86 (3)**, 251–270 (1997). 21

[89] Otero, F. E. B., Silvia, M. M. S., Freitas, A. A., Nievola, J. C., "Genetic programming for attribute construction in data mining", *Genetic Programming*, Springer Berlin Heidelberg, 384–393 (2003). 73

[90] Otero, J., Sánchez, L., "Induction of descriptive fuzzy classifiers with the Logitboost algorithm", *Soft Computing*, **10 (9)**, 825–835 (2006). xi, 28, 29

[91] Pagallo, G., "Learning DNF by decision trees", *In Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, 639–644 (1989). 70

[92] Pérez, R., "Aprendizaje de reglas difusas usando algoritmos genéticos", Doctoral dissertation, Ph.D. thesis, University of Granada, Dpto. Ciencias de la Computación e Inteligencia Artificial (1997). 22

[93] Perzina, R., Ramik, J., "Self-learning genetic algorithm for a timetabling problem with fuzzy constraints", *International Journal of Innovative Computing, Information and Control*, **9 (11)**, 4565–4582 (2013). 111

[94] Qi, J., Cabot, F., Moran, M. S., Dedieu, G., "Biophysical parameter estimations using multidirectional spectral measurements", *Remote Sensing of Environment*, **54**, 71–83 (1995). 103

[95] Quinlan J.R., "C4.5: Programs for Machine Learning", *Morgan Kauffman Publishers* (1993). 140

[96] Rückert, U., Kramer, S., "Margin-based first-order rule learning", *Machine Learning*, **70 (2-3)**, 189–206 (2008). 71

[97] Scherer, R., "Relational modular fuzzy systems", *Studies in Fuzziness and Soft Computing*, **288**, 39–50 (2012). 74

[98] Shaffer, J., "Modified sequentially rejective multiple test procedures", *Journal of American Statistical Association*, **81**, 826–831 (1986). 155

[99] Shafti, L. S., Pérez, E., "MDL-based fitness for feature construction", *In GECCO 2007: Proceedings of the 9th annual conference on Genetic and Evolutionary Computation*, 1875–1875 (2007). 73

[100] Shafti, L. S., Pérez, E., "Reducing complex attribute interaction through non-algebraic feature construction", *In Proceedings of the IASTED International Conference on Artificial Intelligence and Applications*, 359–365 (2007). 73

[101] Sheskin, D., "Handbook of parametric and nonparametric statistical procedures", *Chapman and Hall/CRC* (2003). 155

[102] Sim, K. P., "Artificial Neural Systems", Pergamon Press (1989). 22

[103] Smith, M. G., Bull, L., "Genetic programming with a genetic algorithm for feature construction and selection", *Genetic Programming and Evolvable Machines*, **6 (3)**, 265–281 (2005). 73

[104] Smith, S. F., "A Learning System Based on Genetic Adaptive Algorithms", Ph. D. Thesis, University of Pittsburgh, 1980. 24

[105] Specia, L., Srinivasan, A., Ramakrishnan, G., Das Graas Volpe Nunes, M., "Word sense disambiguation using inductive logic programming", *Lecture Notes in Computer Science*, 4455 LNAI, 409–423 (2007). 71

[106] Sondhi, P.,"Feature Construction Methods: A Survey", sifaka.cs.uiuc.edu (2009). 69, 70, 71

[107] Soua, B., Borgi, A., Tagina, M., "An ensemble method for fuzzy rule-based classification systems", *Knowledge and information systems*, **36(2)**, 385–410 (2013). 112

[108] Stavrakoudis, D. G., Theocharis, J. B., Zalidis, G. C., "A multi-stage genetic fuzzy classifier for land cover classification from satellite imagery", *Soft Computing*, **15**, 2355–2374 (2011). 105, 111

[109] Sugeno, M., Kang, G. T., "Structure identification of fuzzy model", *Fuzzy Sets and Systems*, **28**, 15–33 (1988). 20

[110] Tackett, W. A., "Genetic Programming for Feature Discovery and Image Discrimination", *Proc. Fifth International Conference in Genetic Algorithms*, 303–311 (1993). 79

[111] Takagi, T., Sugeno, M., "Fuzzy identification of systems and its application to modeling and control", *IEEE Transactions on Systems, Man, and Cybernetics*, **15 (1)**, 116–132 (1985). 20, 27

[112] Vafaie, H., De Jong, K., "Genetic algorithms as a tool for restructuring feature space representations", *In TAI 95: Proceedings of the Seventh International Conference on Tools with Artificial Intelligence*, IEEE Computer Society, 8–11 (1995). 72

[113] Venturini, G., "SIA: a supervised inductive algorithm with genetic search for learning attribute based concepts", *In Proc. European Conference on Machine Learning*, 280–296 (1993). 25

[114] Vieira, J., Dias, F. M., Mota, A., "Neuro-fuzzy systems: a survey", *International Conference on Neural Networks and Applications*, Udine (Italia) (2004). 21

[115] Vogelmann, T. C., "Plant Tissue Optics", *Annual Review of Plant Physiology and Plant Molecular Biology*, **44**, 231–251 (1993). 103

[116] Wang, L. and Mendel J.M., "Generating fuzzy rules by learning from examples", *IEEE Transactions on Systems, Man, and Cybernetics*, **22 (6)** (1992). 21, 33

[117] Wehenkel, L., "On uncertainty measures used for decision tree induction", *Proceedings of the Information Processing and Management of Uncertainty on Knowledge-Based Systems IPMU 1996*, 413–418 (1996). 83

[118] Wilcoxon, F., "Individual comparisons by ranking methods", *Biometrics bulletin*, 80–83 (1945). 155

[119] Zadeh, L. A., "The concept of a linguistic variable and its application to approximate reasoning", *Springer US*, 1–10 (1974). 4, 12

[120] Zhang, Y., Hong, G., "An IHS and wavelet integrated approach to improve pan-sharpening visual quality of natural colour IKONOS and QuickBird images", *Information Fusion*, **6**, 225–234 (2005). 105

[121] Zhao, W., Niu, Q., Li, K., Irwin, G. W., "A Hybrid Learning Method for Constructing Compact Rule-Based Fuzzy Models", *IEEE transactions on cybernetics* (2013). 112