Information Fusion 90 (2023) 283–297

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Full length article

The Krypteia ensemble: Designing classifier ensembles using an ancient Spartan military tradition

J. Fumanal-Idocin [a],[*], O. Cordón [b], H. Bustince [a]

[a] Public University of Navarra and Institute of Smart Cities, Campus Arrosadia s/n, 31006 Pamplona, Spain
[b] Deptartment of Computer Science and Artificial Intelligence and Andalusian Research Institute DaSCI, "Data Science and Computational Intelligence", University of Granada, 18071 Granada, Spain

ARTICLE INFO

Keywords:
Classifier ensemble
Optimal classifier selection
Social network
Human social behaviour
Multi-agent systems
Sparta
Krypteia

ABSTRACT

In this work we propose a new algorithm to train and optimize an ensemble of classifiers. We call this algorithm the Krypteia ensemble, based on an ancient Spartan tradition designed to convert their most promising individuals into future leaders of their society. We show how to adapt this ancient custom to optimize classifiers by generating different variations of the same task, each one offering different hardships according to distinct stochastic variables. This is thus applied to induce diversity in the set of individual weak learners. Then, we use a set of agents designed to select those subjects who excel in their assignments, and whose interaction minimizes excessive redundancies in the resulting population. We also study how different Krypteia ensembles can be stacked together, so that more complex classifiers can be built using the same procedure. Besides, we consider a wide range of different aggregation functions in the decision making phase to find the optimal performance for the different Krypteia ensemble variations tested. Finally, we study how different Krypteia ensembles perform for a wide range of classification datasets and we compare them with other state-of-the-art design techniques of classifier ensembles, obtaining favourable results to our proposal.

1. Introduction

Data analysis is one of the most popular disciplines in computer science in the present day [1,2]. There are many problems related to data processing that have been heavily studied due to their academic and economic interest, like data visualization [3], pattern discovery [4–6], and image processing [7] among others. One of the most common tasks in data analysis is to discriminate a series of inputs into a desired category, which is commonly called a classification task [8]. Some of the most popular classification algorithms are those based on neural networks [9], the family of Bayes classifiers [10,11], the K-Nearest neighbours [12], and the Support vector machines [13].

Due to the limitations of these models, a very popular approach to improve classification performance is to form an "ensemble" of classifiers [14–16], which consists of a set of classifiers trained under different configurations that make a decision together [17]. Usually, this consensus is formed by taking the average or the majority vote of these classifiers [18]. There are many classifier ensemble design approaches in the literature. E.g. the random forest trains a set of different decision trees under distinct subsampling conditions and then computes the final decision as the majority vote [19]; Bagging classifiers are formed by training many different subsamples with repeated samples from the original dataset [20]; and AdaBoost iteratively trains different classifiers, each one especially focused on the errors made by the previous classifiers [21]. Of course, since the development of these ensemble construction techniques there has been many research aimed at improving them [22–25]. Some of the most common techniques to improve the performance of an ensemble include using only a subset of the trained classifiers, which is called overproduce-and-choose [26], and using different kinds of classifiers, which is called an heterogeneous ensemble [27].

One of the most researched topics in classification ensembles is the decision making phase, where the majority vote or the arithmetic mean are substituted by another aggregation function [28]. Some of the most popular ones are the Choquet Integral [29], the Sugeno Integral [30], the Ordered Weighted Aggregation operators [31], and the Overlap functions [32]. It is also possible to use other aggregation functions obtained by means of the so-called penalty functions [33].

There are also some interesting research lines in classifier ensemble design focusing on the influence of diversity in the ensemble accuracy [34], and on the trade-off between accuracy and complexity (i.e. selecting the optimal number of classifiers) [35].

Bearing in mind the issues above, the objectives of this work are:

[*] Corresponding author.
E-mail addresses: javier.fumanal@unavarra.es (J. Fumanal-Idocin), ocordon@decsai.ugr.es (O. Cordón), bustince@unavarra.es (H. Bustince).

https://doi.org/10.1016/j.inffus.2022.09.021
Received 8 February 2022; Received in revised form 23 September 2022; Accepted 25 September 2022
Available online 30 September 2022

- To create a new algorithm to train an heterogeneous ensemble using stochastic conditions to adapt the original task to different difficulty settings.
- To study different hierarchical decision making schemes and different aggregation functions to obtain the final solution.

In order to do so, we propose a new approach to train an ensemble of classifiers based on an ancient Spartan ritual called the Krypteia [36]. In this ritual the young noble Spartans would be promoted to adulthood by proving themselves worthy through surviving alone in a hostile land for an unknown period of time. This situation made the participants act in situations of high uncertainty and lack or resources, which spurs heterodox thinking and smart use of resources. Due to the natural ever-changing conditions of real life, not two Krypteia rituals were the same, which also favoured diversity in the way subjects survived the trial, warranting individual fitness and compatible traits for the higher ranks of the Spartan society. This kind of training custom seems to tackle in real-life population some of the problems that are present in modern day classifier ensembles, such as ensemble diversity and accuracy [37], and the trade-off between them [38,39], or robustness against adversarial examples [40].

Following the same idea as in the original Krypteia, our proposed algorithm trains a heterogeneous ensemble using stochastic conditions to adapt the original task to different difficulty settings. Then, we use a novel technique to perform overproduce-and-choose to minimize redundancies in the system. We also study different hierarchical decision making schemes and different aggregation functions to perform the final solution.

Following this strategy, we expect the resulting population to learn different and intelligent solutions, according to each individual subject's situation, and to find good collective solutions when all the subjects' outputs are combined. The goodness of our proposal is shown in a large series of experiments in real-world datasets, comparing the results obtained using different Krypteia ensembles to other state-of-the-art classifier ensemble design methods.

The rest of this paper goes as follows: first, in Section 2 we discuss some relevant works in ensemble design. In Section 3 we recall some concepts of aggregation functions. In Section 4 we explain the ancient ritual of the Krypteia and our proposal to emulate it in a computational environment. Next, in Section 5 we detail the specifics of each step in the design process. Then, in Section 6 we describe the experiments we performed using Krypteia ensembles, and in Section 7 we compare the obtained results to those generated using other types of ensembles. Finally, in Section 9 we give our final remarks and future lines for this work.

2. Related work

Due to their massive popularity and numerous applications, many works have been devoted to enhance performance in classifier ensemble systems. Some of the main lines of research in this topic are model generation for the classifier, model selection for the classifier, and combining the output from the classifiers.

2.1. Individual classifier generation and diversity induction

Ensemble generation is based on learning individual classifiers, weak learners, whose outputs can be combined then into one final output for the global system [41]. The ideal set of classifiers for an ensemble are both accurate and complementary, so that the errors committed by the individual models are corrected in the collective decisions taken.

The most popular ways to generate classifiers are bagging [20], in which each model is trained using a random subsample from the original data; boosting [42], where classifiers are iteratively trained based on the previous errors obtained by the ensemble; and clustering-based approaches [43], where data are clustered according to the different patterns found in the original data and a dedicated classifier is used to classify the samples for each cluster. From this classical approaches, bagging is usually preferred to boosting, as bagging can be computed in a parallelized setting, while boosting requires an iterative process. Clustering based approaches can be very effective when the decision boundaries in the dataset are not constant. However, they are limited by the structures found by the clustering algorithm used. For example, if we used the popular K-Means algorithm, we can only detect convex structures, so complex regions will still be problematic.

Depending on the kind of classifiers used, we denote a homogeneous ensemble, if all the classifiers are of the same type, and a heterogeneous ensemble, if they are different. Many approaches have been studied in both paradigms. In [44] the authors proposed a homogeneous ensemble of neural networks for word classification and a heterogeneous one was used in [45] for a similar problem, using deep learning and classical algorithms as well. Also, in [46] the authors studied heterogeneous ensembles applied to online data streams. Heterogeneous ensembles offer more diversity than their homogeneous counterpart. Nevertheless, the proportions in which the different classifiers should form the ensemble design involves another problem [47]. In [48] the performance of heterogeneous ensembles is compared to homogeneous ones to deal with imbalanced classification problems, finding favourable results to the former ones.

2.2. Classifier selection

As the usefulness of each of the trained classifiers can vary significantly [49], one popular approach in ensemble design is to choose only a subset of classifiers, or to purge a percentage of the classifiers generated [50], which is commonly called overproduce-and-choose (OCS) [26].

Recent works regarding classifier selection and pruning include the proposal in [51] where the authors prune a pool of ensembles based on the indecision region of each classifier. Meanwhile, the proposal in [52] uses the K-means algorithm to cluster the candidates and find the ones that minimize redundancies. Another successful approach to purge classifiers is using meta-features [53]. Meta-features are features extracted from the original data that are used to train meta-classifiers that discriminate between good and bad candidates. Selecting classifiers can be useful when there is a lot of redundancy in their outputs. However, this redundancy can sometimes be useful to minimize the impact of outlier predictions and underperforming classifiers. Meta-feature methods can also tackle this problem by determining which classifiers are good for each sample. However, they also impose additional design problems, like determining which meta-features are good for this task and the boundary conditions to determine if a classifier is competent or not.

Finally, it is also possible to use optimization algorithms to choose the most effective subset of classifiers [54,55] or features [56]. Optimization approaches can lead to good results both in the case of classifier and feature selection. However, they require a proper modelling and a suitable algorithm for this task. Besides, as the validation set used to optimize each configuration is usually only a fraction of the training set, there is also the risk of overfitting.

2.3. Classifier combination

The classifier fusion is commonly performed using functions such as the maximum, the arithmetic mean, and the majority vote [24,57,58]. These functions work best when there is independence between the errors among the classifiers. However, that condition is usually not guaranteed [38].

A common solution to this problem is to ponder each classifier according to its importance [59]. Other relevant approaches include the use of fuzzy aggregations to model uncertainty and coalitions of the inputs to fuse. Some popular operators in this phase include the Choquet and Sugeno integrals [60–62]. These operators can learn the interaction among the features and take them into account when aggregating the data. They are also simpler to use compared to other pruning or feature selection mechanisms. However, they also require to learn a suitable fuzzy measure to capture these coalitions, which is not straightforward [63]. Another successful strategy consist of using fuzzy linguistic rule-based classification system as the fusion process, so that it can be interpretable for the user [64].

It is also possible to fuse the classifiers in different phases, using a hierarchical fusion phase [18,27], which can also include different aggregation operators [60]. This procedure can achieve higher accuracy results than fusing all the classifiers in one phase. However, it requires a sensitive hierarchy system among the classifiers and a suitable aggregation operator in each phase. Choosing the best aggregation operator for one vector can be solved using a penalty or a moderate deviation [65], but the interaction among aggregations in a hierarchical fashion has not been studied.

3. Preliminaries

In this section we shall recall some of the concepts related to the most common aggregation functions, the formulation of the Choquet and Sugeno integrals, and some of their generalizations, the Overlap functions and the Ordered Weighted Aggregation operators.

3.1. Properties of aggregation functions

Aggregation functions are used to fuse information from n sources into one single output. A function A: $[0,1]^n \to [0,1]$ is said to be a n-ary aggregation function if the following conditions hold:

- A is increasing in each argument: $\forall i \in \{1,\dots,n\}$, if $x_i < x_j$, $A(x_1,\dots.,x_i,\dots x_n) \le A(x_1,\dots,x_j,\dots x_n)$. For example, consider the vectors $\mathbf{x} = \{0.3, 0.9, 0.1, 0.8\}$ and $\mathbf{z} = \{0.3, 0.9, 0.6, 0.8\}$. If A is an aggregation function, it must hold that $A(\mathbf{x}) \le A(\mathbf{z})$, because all elements of in $\mathbf{x}$ are equal than those in $\mathbf{z}$, except for the case of x_3 and z_3, where $0.1 < 0.6$.
- $A(0,\dots,0) = 0$
- $A(1,\dots,1) = 1$

Some examples of classical n-ary aggregation functions are:

- Arithmetic mean: $A(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} x_i$.
- Max: $A(\mathbf{x}) = max(x_1,\dots,x_n)$.
- Min: $A(\mathbf{x}) = min(x_1,\dots,x_n)$.

3.2. T-norm

A T-norm is an aggregation function $[0,1]^2 \to [0,1]$ that satisfies the following properties for $x, y, z \in [0,1]$ [66]:

- $T(x,y) = T(y,x)$
- $T(x,T(y,z)) = T(T(x,y),z)$
- $T(x,1) = x$

Some examples of T-norms are the product, the minimum and the Łukasiewicz T-norm:

$$L_{luk}(x,y) = \max(0, x+y-1) \tag{1}$$

3.2.1. Overlap functions

An n-dimensional overlap is an aggregation function $G : [0,1]^n \to [0,1]$ such that [32]:

- G is commutative.
- $\prod_{i=1} x_i = 0$ if and only if $G(\mathbf{x}) = 0$.
- $\prod_{i=1} x_i = 1$ if and only if $G(\mathbf{x}) = 1$
- G is increasing.
- G is continuous.

Some examples of overlap functions are:

- Minimum: $G(\mathbf{x}) = \min(\mathbf{x})$
- Harmonic Mean (HM): $G(\mathbf{x}) = \frac{n}{\sum_{i=1}^{n}\frac{1}{x_i}}$
- Sinus Overlap (SO): $G(\mathbf{x}) = \sin(\frac{\pi}{2}\Pi_{i=1}^{n} x_i)$
- Geometric Mean (GM): $G(\mathbf{x}) = \sqrt[n]{\prod x_i}$

3.3. Ordered Weighted Averaging operators (OWA)

$\mathbf{w} = (w_1,\dots,w_n) \in [0,1]^n$ is called a weighting vector if $\sum_{i=1}^{n} w_i = 1$. The OWA operator associated to $\mathbf{w}$ is the mapping $\text{OWA}_{\mathbf{w}} : [0,1]^n \to [0,1]$ defined for every $\mathbf{x} = (x_1,\dots,x_n) \in [0,1]^n$ by [67]:

$$OWA(\mathbf{x}) = w_1 x_{\gamma(1)} + \dots + w_n x_{\gamma(n)} \tag{2}$$

where $\gamma : \{1,\dots,n\} \to \{1,\dots,n\}$ is a permutation such that: $x_{\gamma(1)} \ge x_{\gamma(2)} \ge \dots \ge x_{\gamma(n)}$.

The weighting vector can be computed used a quantifier function, Q. For this study, we have used the following one:

$$w_i = Q_{a,b}(\frac{i}{n}) - Q_{a,b}(\frac{i-1}{n}) \tag{3}$$

$$Q_{a,b}(x) = \begin{cases} 0, & \text{if } x < a \\ 1, & \text{if } x > b \\ \frac{x-a}{b-a}, & \text{otherwise} \end{cases} \tag{4}$$

where $a, b \in [0,1]$ and $a < b$. Depending on the value of the parameters a and b, different weight vectors can be obtained. We have considered three different ones:

- OWA_1: $a = 0.1, b = 0.5$
- OWA_2: $a = 0.5, b = 1$
- OWA_3: $a = 0.3, b = 0.8$

3.4. Choquet integral

Having $N = \{1,\dots,n\}$, a function $m : 2^N \to [0,1]$ is a fuzzy measure if, for all $X, Y \subseteq N$, it satisfies the following properties [28]:

($m1$) Increasingness: if $X \subseteq Y$, then $m(X) \le m(Y)$.
($m2$) Boundary conditions: $m(\emptyset) = 0$ and $m(N) = 1$.

The discrete Choquet integral with respect to m is defined as the function $C_m : [0,1]^n \to [0,1]$ given for every $\mathbf{x} = (x_1,\dots,x_n) \in [0,1]^n$, by:

$$C_m(\mathbf{x}) = \sum_{i=1}^{n}(x_{\sigma(i)} - x_{\sigma(i-1)}) \cdot m(A_i) \tag{5}$$

where $\mathbf{x}_\sigma$ is an increasing permutation of $\mathbf{x}$ such that $0 \le x_{\sigma(1)} \le \dots \le x_{\sigma(n)}$. With the convention that $x_{\sigma(0)} = 0$, and $A_i = \{(i),(i+1),\dots,(n)\}$.

Two important generalizations of the Choquet integral are the CF [61] the $C_{F1,F2}$ [68] integrals, and the d-Choquet integrals, in which the difference between the inputs in Eq. (5) is changed by a dissimilarity [69].

3.4.1. CF integral

The CF is a generalization of the Choquet integral that replaces the product used in Eq. (5) for a more general function F. In [70] the authors detail the required properties for F so that the CF is an aggregation or a pre-aggregation function, and conclude that the best F in their experimental results is the Hamacher T-norm. For this reason, we have chosen it for our experimentation, as detailed in the following expressions:

$$T_H(x,y) = \begin{cases} 0, & \text{if } x = y = 0 \\ \frac{xy}{x+y-xy}, & \text{otherwise} \end{cases} \tag{6}$$

$$CF_m(\mathbf{x}) = \sum_{i=1}^{n} T_H(x_{\sigma(i)} - x_{\sigma(i-1)}, m(A_i)) \tag{7}$$

3.4.2. $C_{F1,F2}$ integral

The original product of the Choquet Integral can be decomposed on two product functions using the distributive property of the product. Therefore, the Choquet integral can be written as:

$$C_m(\mathbf{x}) = \sum_{i=1}^{n} x_{\sigma(i)} m(A_i) - x_{\sigma(i-1)} m(A_i) \tag{8}$$

Then, the product functions are substituted for two more generic functions: F_1 and F_2. In [68] the authors explain the properties that must hold F_1 and F_2 so that the $C_{F1,F2}$ is an aggregation or a pre-aggregation function. Consequently, the expression for the $C_{F1,F2}$ is the following:

$$C_{F1,F2}(\mathbf{x}) = \sum_{i=1}^{n} F_1((x_{\sigma(i)}), m(A_i)) - F_2((x_{\sigma(i-1)}), m(A_i)) \tag{9}$$

For our experimentation, we take $F_1 = \sqrt{xy}$ and F_2 is the Łukasiewicz T-norm.

3.5. Sugeno integral

Let $m : 2^N \to [0,1]$ be a fuzzy measure. The discrete Sugeno integral with respect to m is defined as a function $S_m : [0,1]^n \to [0,1]$, given for every $\mathbf{x} = (x_1, \ldots, x_n)$ [71], following the same notation as in the Choquet integral in Eq. (5):

$$S_m(\mathbf{x}) = \max\{min(x_{\sigma(i)}, m(A_i)) | i = 1, \ldots, n\} \tag{10}$$

Two generalizations of the Sugeno Integral are the Hamacher-based Sugeno integral and the FG-Sugeno.

3.5.1. Hamacher-based Sugeno integral

If we consider using the Hamacher T-norm instead of the minimum in Eq. (10), we obtain the following expression [72]:

$$S_m^{TH}(\mathbf{x}) = \max\{T_H(x_{\sigma(i)}, m(A_i)) | i = 1, \ldots, n\} \tag{11}$$

3.5.2. FG-Sugeno

If we replace the minimum for the product, and the maximum for the sum in Eq. (10), we obtain the following expression [62]:

$$S_m^{FG}(\mathbf{x}) = \sum_{i=1}^{n} (x_{\sigma(i)} m(A_i)) \tag{12}$$

4. The Krypteia ensemble

The Krypteia ensemble is a novel classifier ensemble algorithm designed to maximize the effectiveness of each individual subject and the variability and performance of their outputs when combined. It does so by mimicking the ancient rite of Krypteia in the ancient Sparta, designed to train the future elites of the Spartan army and government.

This algorithm consists of three main steps:

1. Survival ordeal: in this phase we train each classifier individually. In order to induce diversity in the training process, we modify the training task for each one in a stochastic process, so that some of them have a easier or harder task than the original classification task. We discard all the classifiers that did not meet the expected accuracy rate in their own task.
2. Social ordeal: this second phase follows an OCS scheme in which we discard those classifiers that have a very similar output to other classifiers with higher accuracy rate. The aim is to avoid having samples with much more weak learners classifying them correctly than others.
3. Aggregation learning: we learn which is the most appropriate function to combine the output of all the surviving classifiers.

The resulting population from this process is called a Krypteia Unit.

A general scheme of the Krypteia algorithm is displayed in Fig. 1. The following subsection briefly describes the Krypteia ritual in ancient Sparta design method and Section 4.2 describes each step of the Krypteia ensemble in detail.

4.1. Krypteia in the ancient Sparta

The Krypteia was a particularly brutal initiation rite in the Spartan society for the young men of the higher ranks of the state [36]. According to Plutarch [73], each year the young noblemen of Sparta would declare war to the Helot population of Sparta, so that any killing or robbery committed was not considered crime. Armed with nothing but a knife, the young Spartans were left alone and sent out in the night to the Helot settlements. They were supposed to obtain their own methods of survival by stealing and killing in their circumvent area.

The origins and exact purpose of the Krypteia are still under debate. This ritual was supposed to be a form of terrorizing and subjugating the Helot population, alongside training the next generation of Spartan leaders, as no young man could aspire to hold positions of power if he had not passed through this ordeal. It is also believed that the Krypteia participants could have been organized as a unit in the Spartan army. In any case, such brutal practices seemed to be effective, as Sparta made a place for itself in history thanks to is great military capacity, and as long as the Krypteia took place, the Helot population stayed under their rule.

It is believed that the Krypteia was disbanded in the battle of Sellasia, in 222 BC, where Sparta lost against the Macedonian army commanded by Antigonus III [74]. This resulted in the emancipation of many helots and without a Helot population, it was impossible to organize the Krypteia.

4.2. Krypteia ensemble: bringing the ancient ritual to modern computational systems

In this section we detail both the Survival and Social ordeals, and how to combine their outputs.

4.2.1. Survival ordeal (Fig. 1-Step 2)

The Survival ordeal is a diversity induction process that consists of randomly modifying the original task in order to obtain a different version of the weak learner. The aim is that this new task should be more difficult than the original most of the times, so that the resulting Krypteia Unit will have classifiers adapted to harder or extreme situations compared to the original, single classifier.

We alter the tasks using two different techniques: sampling modifications, denoted as “Data ordeal”, and prediction alterations, which we called “Bias ordeal”. The Data ordeal consists of performing an undersampling, data augmentation, or oversampling technique of one or various classes in the classification task. The Bias ordeal consists of adding an artificial bias to one or more classes to the actual output of the classifier. Whether or not a classifier is affected by the Data ordeal or the Bias ordeal, the extent of that ordeal and the number of classes affected, is all decided randomly.

Once the Data ordeal and Bias ordeal have been completed we evaluate the performance of the altered classifier. If this performance is less than a survival threshold, we discard that classifier. Following the Krypteia metaphor, this means that the soldier was not strong enough to survive the ritual and was killed by the Helots or the environment.

The ordeals can be too hard, so that all classifiers fail. It might also happen that the number of surviving classifiers is higher than the expected value. So, it is important to take into consideration the natural difficulty of the original classification task in order to establish the survival threshold, or to establish additional measures to guarantee that a reasonable number of weak learners survive the Survival ordeal.

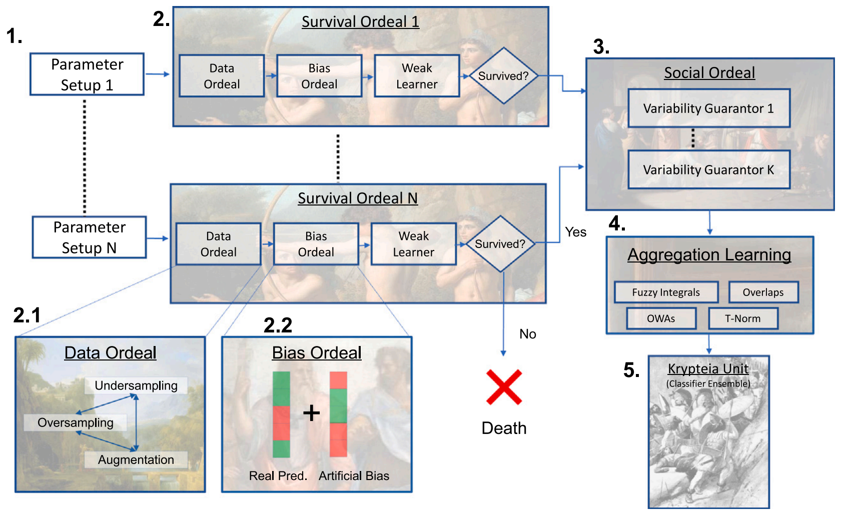

Fig. 1. Visual scheme of the Krypteia algorithm. **1.** We generate random sets of parameters. **2.** Each of these parameter settings creates a different Survival ordeal, where the weak learners need to correctly solve a stochastically modified version of the original classification task. The modification includes the Data ordeal (**2.1**), where we manipulate the training data, and the Bias ordeal (**2.2**), where we compute a random vector that we will add as an artificial bias to each of the classifiers predictions. **3.** If they pass the Survival ordeal, a selection of the surviving classifiers is performed by k Variability Guarantors in the Social ordeal to minimize redundancies. **4.** We choose the best aggregation function for the decision making phase of the ensemble. **5.** The output of the Social ordeal is the Krypteia Unit. Section 4.2 contains a detailed description of each component behaviour.

4.2.2. *Social ordeal (Fig. 1-Step 3)*

Once the individual classifiers have been proved by the Survival ordeal, we make the population of classifiers pass through the Social ordeal, which is an OCS scheme that measures how diverse is the “society” formed by these classifiers and removes those that produce redundant results with respect to the rest of the population.

The idea is that, although the variability within the Survival ordeal itself makes the classifiers different, there could be redundancies, especially if the survival rate is high. For example, given 0 as a complete failure and 1 as a total success in the Survival ordeal, if we set the survival threshold to 0.95, then we are forcing all the surviving classifiers to present very similar outputs.

To solve this problem, in the Social ordeal we create a set of n Variability Guarantors (VGs), whose aim is to guarantee the diversity in the output of each classifier in the final Krypteia Unit. To do so, each VG subsamples a very small fraction of the input data for each subject (1%, for example) and then checks how each classifier performs for this data split.

Each VG marks as “good” the subjects that correctly classified its subsampled data. Since each VG takes only a small fraction of the input data, the intersection of the data chosen among the VGs will be virtually null. In this way, we avoid having in the final Krypteia Unit a long list of classifiers that performed very well in the same subset of data, whilst performed very poorly on other. Besides, since the VGs uniformly sample the dataset, their subsampled data can contain both easy and hard to correctly classify observations. In this way, VGs reject those classifiers that performed well only on the easy set of observations, even if they obtained a higher accuracy rate than the classifiers marked as good by the VGs.

4.2.3. *Decision making*

The most straightforward way to make a decision with a Krypteia Unit is to fuse the output of each weak learner using any of the aggregation functions presented in Section 3. However, although a Krypteia Unit can obtain good results on its own, the stochastic nature of the training process can result in many different outcomes. This is one of the strengths of the Krypteia, but it can also result in poor performance in some cases. To minimize the negative impact of the possible faulty units, we propose to stack different Krypteia Units in different decision making phases, in a hierarchical decision making scheme. This proposal has another benefit: we can use different aggregation functions in each level, which can result in better performance.

The different hierarchical schemes that can be used are illustrated in Fig. 2. A decision from a set of N Krypteia units can be obtained in three different ways:

1. Unit-all: we fuse the output from all weak learners within the Krypteia units in one phase, using just one aggregation function.
2. Division-all: we denote as a Krypteia Division the appending of the output of different Krypteia Units. Then, the Division-all scheme consists of fusing the output of various Krypteia Divisions obtained from the N Krypteia units, using one aggregation function in each Krypteia Unit and another for the Krypteia divisions. Krypteia units are assigned to different Krypteia divisions randomly.
3. Krypteia army: consists of obtaining the output for every Krypteia Division individually and then fusing their outputs, using a total of three aggregation functions.

5. Training a Krypteia Unit for a classification problem

In this section we focus on how to implement the following stages of a Krypteia Unit training process:

1. How to setup the parameters.
2. How to perform the Survival ordeal, and how difficult it should be.
3. How to configure the Social ordeal.
4. How to choose the aggregation function to make the decision.

5.1. *Parameter setup*

For the case of a classification system, a Krypteia consists of a set of weak learners. However, we do not specify each one individually; instead, we set a list of parameters for the whole Krypteia Unit (see Table 1 for example) in order to induce diversity within each unit. These parameters are:

1. The number of subjects wanted in the Krypteia Unit.
2. The proportion of each type of classifiers.

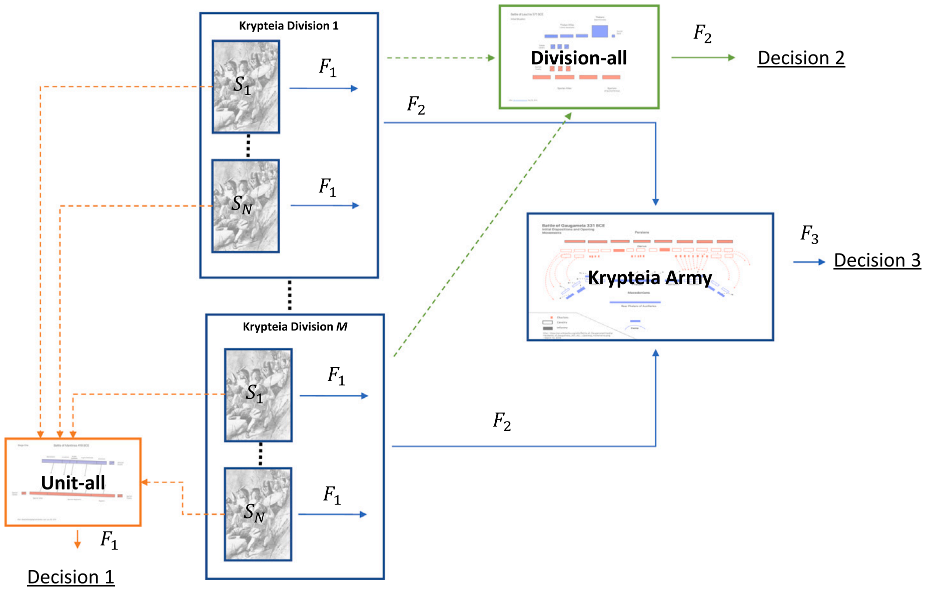

Fig. 2. Visual scheme showing the algorithm proposed in Section 4.2.3 to generate and combine the output of multiple Krypteia Units. A Krypteia Division is obtained by appending the output from N Krypteia Units and a Krypteia Army by appending and fusing the output of M Krypteia Divisions. A decision from a set of N Krypteia Units can be obtained in three different ways: **Decision 1**: consists of fusing all the weak learners within the Krypteia Units in one phase, using just one aggregation function. This scheme is denoted "Unit-all". **Decision 2**: consists of appending the output of the Krypteia Divisions, using one aggregation function to fuse each Krypteia Unit and another for the Divisions. This scheme is called "Division-all". **Decision 3**: consists of fusing every Krypteia Division individually and the fusing the output from all of them, using a total of three different aggregation functions. This scheme is the Krypteia Army. Images taken from [75].

Table 1
Example of two Krypteia Unit configurations. For each unit, we specify the number of classifiers wanted in the final population and the proportion of each kind, the number of VGs, the minimum accuracy needed to pass the Survival ordeal, and the ordeals performed in this unit.

N	KNN	LDA	QDA	SVM	Tree	VGs	Survival Thr.	Ordeal
24	0.08	0.12	0.20	0.12	0.25	6	0.5067	Survival and Social
7	0.14	0.14	0.14	0.28	0.28	–	0.9266	Survival

3. The number of VGs in the Social ordeal.
4. The initial Survival Threshold.
5. Which Ordeals to perform.

By setting different parameters for each unit, we can maximize the diversity between various Krypteia Units trained for the same task. This can be of use, for example, if we want to train different Krypteia Units and then choose the best one, or if we want to use the Krypteia Division or a Krypteia Army decision schemes.

For our tests, we have worked with five different types of classifiers:

- K-Nearest Neighbours (KNN) ($K = 6$ for this experimentation) [12].
- Linear Discriminant Analysis (LDA) [76].
- Quadratic Discriminant Analysis (QDA) [77].
- Support Vector Machines (SVM) [13].
- Decision trees [78].

We choose a random proportion for each type of classifier, a random number between 1 and 10 for the number of VGs, and a random number between 0 and 1 for the Survival Threshold. The type of Ordeal can be: Survival only, Social only, or both, each of the possibilities with the same probability.

5.2. Designing the Survival ordeal

The Survival ordeal consists of two complementary ordeals: the Data ordeal and the Bias ordeal, in which we artificially modify the weak learner behaviour. Once both ordeals have been stated, we also need to adjust the Survival Threshold so that the survival rate is within acceptable boundaries.

5.2.1. Data ordeal

The Data ordeal consists of modifying the training data for a weak learner. We can do so by subsampling the data, which theoretically makes the problem harder, by using oversampling techniques, which should make the problem easier for the classifier, or by using both. In that case we can also differ in which order both procedures are applied. The idea is not to necessarily obtain a stratified dataset, which is not guaranteed to happen, but to obtain maximum variation in the different synthetic training sets generated.

We have used the following techniques to perform the sampling manipulations:

- Random undersampling: which consists of randomly sampling a percentage of the original data.
- Centroid Undersampling (CU): we use the K-means algorithm to compute k centroids, being k the number of samples wanted from the original data.
- Synthetic Minority Over-sampling TEchnique [79] (SMOTE): generates new samples by interpolating examples from the same class.
- SMOTE + Tomek Links (SMOTETomek) [80]: combines the oversampling of SMOTE and Tomek Links to delete overlapping observations between classes.
- SMOTE + Wilson's Edited Nearest Neighbour Rule (SMOTEEN) [80]: combines the oversampling of SMOTE with undersampling based on the Edited Nearest Neighbour Rule [81].

In order to determine how much each event is going to happen, we set a parameter p. The lower this p, the more likely is that a data manipulation happens.

The scheme the followed by the Data ordeal is:

1. Draw a random number. If this number is bigger than p, then the Data ordeal takes place (step 2); if not, then there are no data manipulations (step 3).

2. Draw a random number. If this number is bigger than 0.5, then we proceed to subsample the data and then oversample it (step 2.1). If not, first we perform the oversampling and then undersampling (step 2.2).
    2.1. Draw a random number. If this number is bigger than p, then we perform the undersampling (Alg. 2). Draw a random number again, and if is bigger than p, we perform the oversampling (Alg. 1).
    2.2. Draw a random number. If this number is bigger than p, then we perform the oversampling (Alg. 1). Draw a random number again, and if is bigger than p, we perform the undersampling (Alg. 2).
3. We train the weak learner on the resulting data.

Algorithm 1: Generating Data Advantage in the Data ordeal

```
Result: $new\_data$
Input: $train\_data$, $p$
$lucky\_number = random()$
if $lucky\_number > p$ then
    if $lucky\_number < p + 1/3 * (1 - p)$ then
        $new\_data = SMOTETomek(train\_data)$
    else if $lucky\_number < p + 2/3 * (1 - p)$ then
        $new\_data = SMOTEEN(train\_data)$
    else
        $new\_data = SMOTE(train\_data)$
    end
end
```

Algorithm 2: Undersampling in the Data ordeal

```
Result: $new\_data$
Input: $train\_data$, $p$
$lucky\_number = random()$
if $lucky\_number > p$ then
    if $lucky\_number < p + 1/2 * (1 - p)$ then
        $new\_data = random\_undersample(train\_data)$
    else
        $new\_data = centroid\_undersample(train\_data)$
    end
end
```

5.2.2. Bias ordeal

The Bias ordeal consists of inducing an artificial bias in the weak learners predictions. The process follows a similar procure to the Data ordeal. We draw a random number, if that number is higher than p, the Bias ordeal takes place, if not, no bias is induced.

To perform the bias induction we draw C random numbers from a uniform distribution in the $[0, 1]$ interval, where C is the number of classes in the classification task, obtaining the vector r_C. When the classifier predicts the probability for each class for a sample, we add the r_C vector to those probabilities.

Although the aim of this ordeal is to augment the variance among weak learners by reducing the individual accuracy of each one, we also have found cases where this artificial bias also improved the performance of individual weak learners. This might be because under the right circumstances, like a very imbalanced training set, this artificial bias can work as a regularization factor.

5.2.3. Setting the Survival Threshold

The Survival Threshold quantifies the level of exigence that we impose to the weak learners after they passed their Survival ordeal in order to accept them as valid weak learners. It is expressed as the minimum accuracy rate required to obtain in the original training set after passing the Data and Bias ordeals. It is a very important factor for the final Krypteia Unit: if it is too low, the resulting classifiers can output almost random predictions; if it is too hard, no classifier will come out. However, this value cannot be interpreted as the ordeal difficulty or the percentage of survival rate. A Survival Threshold of 0.9 can be too easy for a classification task that any individual classifier can perform almost perfectly, and a Survival Threshold of 0.5 can be too much if the learning task is very difficult.

The solution to this problem is an adaptative threshold, i.e to start with a random Survival Threshold, and then decrease it if the death rate is too high. We reduce the Survival Threshold by 0.05 each time we register as many deaths as twice the number of classifiers in the Unit until the desired number of classifiers is obtained. That mimics the robust approaches followed by metaheuristics to adapt their main diversification–intensification trade-off parameter [82].

5.3. Configuring the Social ordeal

The VGs compose the Social ordeal. They are supposed to convert a pack of good individual classifiers into a truly functional ensemble of classifiers. Two parameters are important to form a collective of VGs that perform a meaningful Social ordeal:

- VG exigence: this is the percentage of the original data that the VG will use to evaluate each weak learner. The lower this parameter is, the more strict each VG will be. This might present a problem if the VGs are too strict, because if the number of test samples is too low, then many of the classifiers might get all of them right, or on the contrary, fail them all. If we choose a value that is too high, then the VGs will have a significant intersection among them, which would also make the diversity search futile. We have chosen value 1% as it seems to correctly balance both issues and avoids oversaturation of one VG.
- Number of VGs: the more VGs, the more subjects will be marked as “good”. We have opted to draw a distribution between 1 and 10, so that with a exigence of 1%, the VGs will sample around $1-10\%$ of the original data.

5.4. Choosing the aggregation function

Once all the subjects have passed through all the ordeals, the final population forms a Krypteia Unit ensemble. This ensemble of classifiers needs to fuse the output of each individual weak learner to reach a final decision. We have studied the functions mentioned in Section 3.1 as possible aggregation functions for the Krypteia.

In order to discriminate the best aggregation function, we have opted for performing a 5-fold validation in the training set for each one of them, and choose the aggregation that resulted in a best accuracy according to this evaluation criteria. In the case of the Krypteia Division and the Krypteia Army, we have tested using all possible combinations of aggregation functions in each phase.

6. Experimental results for different Krypteia ensembles

In this section we study the performance of the Krypteia ensemble for a wide range of classification datasets. We show the performance for different decision making strategies, a study of the importance of the different parameters of a Krypteia Unit, and how the Krypteia ensembles compare to other kinds of classifier ensemble design approaches.

6.1. Datasets studied in the experimentation

For our experimentation, we used a large set of 52 different classification datasets, obtained from the Keel database [83]. The number of samples and attributes for each one are reported in Table 2. We take a standard 80/20 training-test partition for each dataset. The metric used to measure the performance is the standard classification accuracy.

Table 2
Datasets used for our experimental results.

Dataset	Instances	Features	Classes	Dataset	Instances	Features	Classes
abalone	4173	8	28	nursery	12 959	8	5
appendicitis	105	7	2	optdigits	5619	64	10
australian	689	14	2	page-blocks	5471	10	5
balance	624	4	3	penbased	10 991	16	10
banana	5299	2	2	phoneme	5403	5	2
bands	364	19	2	pima	767	8	2
breast	276	9	2	post-operative	86	8	3
bupa	344	6	2	ring	7399	20	2
car	1727	6	4	saheart	461	9	2
chess	3195	36	2	satimage	6434	36	6
coil2000	9821	85	2	segment	2309	19	7
contraceptive	1472	9	3	sonar	207	60	2
crx	652	15	2	spambase	4596	57	2
dermatology	357	34	6	spectfheart	266	44	2
ecoli	335	7	8	splice	3189	60	3
flare	1065	11	6	texture	5499	40	11
german	999	20	2	thyroid	7199	21	3
haberman	305	3	2	tic-tac-toe	957	9	2
housevotes	231	16	2	titanic	2200	3	2
ionosphere	350	33	2	twonorm	7399	20	2
iris	149	4	3	vehicle	845	18	4
letter	19 999	16	26	vowel	989	13	11
magic	19 019	10	2	wdbc	568	30	2
mammographic	829	5	2	wine	177	13	3
mushroom	5643	22	2	wisconsin	682	9	2
newthyroid	214	5	3	yeast	1483	8	10

6.2. Results for the Krypteia Unit

In this section we have studied the results using 300 Krypteia Units for each dataset, trained using the procedures in Section 5. We chose that number to honour the famous Battle of Thermopylae between Spartans and Persians [84].

Table 3 collects the results for the Krypteia Units in each one of the datasets considered, divided into three columns. In the first column, we show the average of all the Krypteia Units' performance. In the second column, we show the results that the best Krypteia Unit obtained (the one with better average accuracy for all the datasets), and finally in the third column we show the best results obtained by a Krypteia Unit in that particular dataset.

In most of them, the difference between the average result and the best Krypteia Unit is not significant, which shows that the Krypteia training process is indeed capable of generally generate good results. However, in some datasets, for example in "letter", there is a +30% accuracy difference between the average and the best unit. This shows that the Krypteia training process is also capable to create vastly superior subjects than the average Krypteia Unit.

In Table 4 we show the average for each column in Table 3, in order to obtain the average performance for all Krypteia Units, the average performance for the best Krypteia Unit, and the average best result over all the datasets considered. We can see here that the best unit generated is clearly superior to the average performance of the rest of the units, and that there is still an average of 2% of difference between the optimal result and the best unit obtained.

6.3. Feature importance in the Krypteia ensemble

As many parameters as the Krypteia Unit has, it is natural to think that some of them are more important than others. In order to compute the importance of each feature, we built a classification and regression tree (CART). The CART model have been used widely in medicine as a way to measure the effects of different treatments in the outcome of a patient. This algorithm trains a random forest with the parameters of a model as inputs and the final outcome of the model as labels [85]. It predicts the performance of a model based on its parameters. Then, we can study the coefficients that the CART model applied to each factor to make its prediction, to learn how they affect the performance of the studied system.

In Table 5 we displayed the results for this analysis, illustrating the Krypteia Units performance. From this table we can infer that the most important factor is the Survival Threshold by a large margin, followed by the Unit size. The proportion of different classifiers seems to be equally important, although using trees seemed to be less important than the rest. The kind of ordeal performed seems not to have much effect.

6.4. Results for the Krypteia Division

In this case, we used the same Krypteia Units as in Section 6.2, but instead of studying the performance of each individual Spartan unit, we followed the process detailed in Section 4.2.3 to stack different Krypteia Units. Table 6 collects the results for each dataset for the average of all Krypteia Divisions, the best Krypteia Division, and the best result obtained by a Krypteia Division for each one.

In Table 7 we show the average of all columns in Table 6. In this case, we obtained a lesser difference between the best and the average result compared to the unit results. This seems to indicate that stacking the units in this way mitigates the impact of bad units, although the best possible result is inferior to that obtained using Krypteia Units.

6.5. Results for the Krypteia Army

In this case we have stacked the Krypteia Units forming one Krypteia Army. We used the same Krypteia Divisions as in Section 6.4 in order to directly compare the effect of an additional level of complexity.

We display the results obtained with the Krypteia Army in Table 8 and the correspondent aggregation trio that achieved that result. We can observe that even though the arithmetic mean is most of the times the chosen aggregation, studying additional aggregation functions allows in many cases to improve the result.

6.6. Comparison of the different Krypteia ensembles

In Table 9 we show the comparison for the average performance of the different Krypteia ensembles: the best Krypteia Unit generated, the best Krypteia Division generated, and the Krypteia Army. We found

Table 3
Results for the Krypteia Units.

Dataset	Avg.	Best unit	Best Res.	Dataset	Avg.	Best unit	Best Res.
abalone	99.73	100.00	100.00	nursery	89.83	99.96	100.00
appendicitis	94.67	95.24	100.00	optdigits	93.80	93.51	98.84
australian	81.57	87.68	88.41	page-blocks	38.61	96.99	97.44
balance	99.78	100.00	100.00	penbased	87.53	97.31	99.49
banana	88.02	90.09	90.47	phoneme	88.54	88.90	90.56
bands	99.34	94.52	100.00	pima	99.64	100.00	100.00
breast	99.44	100.00	100.00	post-operative	39.69	100.00	100.00
bupa	67.35	68.12	78.26	ring	91.11	93.72	98.51
car	99.77	100.00	100.00	saheart	99.97	100.00	100.00
chess	99.96	100.00	100.00	satimage	89.55	90.37	92.15
coil2000	93.54	94.50	94.66	segment	97.29	97.40	98.92
contraceptive	53.22	54.92	58.64	sonar	99.85	100.00	100.00
crx	99.62	100.00	100.00	spambase	92.11	92.72	93.80
dermatology	86.84	98.61	100.00	spectfheart	74.93	79.63	85.19
ecoli	96.68	98.51	100.00	splice	98.41	100.00	100.00
flare	99.77	100.00	100.00	texture	94.31	94.55	99.91
german	89.91	100.00	100.00	thyroid	98.60	99.79	100.00
haberman	99.99	100.00	100.00	tic-tac-toe	99.99	100.00	100.00
housevotes	99.72	100.00	100.00	titanic	72.44	80.45	82.27
ionosphere	99.94	100.00	100.00	twonorm	91.11	86.01	98.37
iris	100.00	100.00	100.00	vehicle	99.39	99.41	100.00
letter	69.81	99.72	100.00	vowel	88.92	90.40	97.98
magic	100.00	100.00	100.00	wdbc	99.97	100.00	100.00
mammographic	80.06	81.33	85.54	wine	99.01	100.00	100.00
mushroom	100.00	99.82	100.00	wisconsin	93.34	93.43	97.08
newthyroid	97.88	100.00	100.00	yeast	98.77	99.33	100.00

Table 4
Performance summary for the Krypteia Units.

	Average units	Best unit	Best result
Average Acc.	87.48	94.09	**96.02**

Table 5
Feature importance for the Krypteia ensemble computed using a CART model.

Feature	% of importance
Nº of KNN	7.47%
Nº of LDA	7.61%
Nº of QDA	7.40%
Nº of SVM	8.19%
Nº of trees	5.13%
Unit size	11.55%
VG sample size	7.02%
Survival threshold	39.95%
Full ordeal	2.00%
Survival-only ordeal	1.57%
Social-only ordeal	2.08%

the Krypteia Army to beat the other two by a 1.5% points margin. We have also computed a homogeneous version of the Krypteia army using only SVM classifiers and the arithmetical mean as the aggregation function. This version of the Krypteia performed worst than the rest of the Krypteia configurations, reinforcing the idea that both the heterogeneity and the aggregation functions improve the ensemble performance.

6.7. Results for unit-all and division-all fusion schemes

Instead of performing the decision making scheme of the Krypteia Army, we can fuse all the 300 units using the unit-all or the division-all fusion schemes.

In Fig. 3 we studied the expected performance for the 300 units in the Krypteia Army studied in Section 6.5 using the division-all fusion scheme (fusing n different units as if they were just one division). To do so, we have computed the average accuracy for all the datasets using up to 300 Krypteia Units. These Krypteia Units are the same as those used in previous sections, selected randomly. There we can see that this system seems to reduce performance beyond $n = 10$, and drops significantly when $n > 50$, stabilizing around 92.40%. In no case this scheme performed better than the Krypteia Army.

In Fig. 4 we performed the same experiment using the unit-all (fusing n different subjects as if they were just one unit) decision making phase. A similar situation happens in this case, where a 20–30% of the original army seems to perform greatly when fused in this way, but then stabilizes in an inferior performance. Again, in no case the average accuracy obtained was better than the Krypteia Army.

6.8. Performance for each aggregation function

Each time we use a Krypteia ensemble, we need to choose among one of many aggregation functions. Based on the results from our experiments, we counted the number of times each one of them was the best performing aggregation. By profiling each one the aggregations, we hope to discard the worst ones in future trainings and to reduce the number of possible candidates.

In Fig. 5 we show the times that each of the different aggregations provided the best result in an experiment. We found the classical arithmetic mean to be the one that won most of the times, but with only a very small margin with respect to the maximum, the minimum, the OWA operators, the n-overlap functions, and the Choquet integral. Sugeno integrals performed poorly and the generalizations of the Choquet integral never won in any case. However, it is worth pointing out that the performance of the fuzzy integrals depends on the fuzzy measure used.

7. Comparing the Krypteia ensemble with other ensemble classifiers

In this section we compared the results obtained with the Krypteia ensemble with seven very diverse types of classifier ensembles [86]. Three of them are classical ensemble algorithms: adaboost, bagging, and majority vote using SVMs and random forests. The hyperparameters for these algorithms were fixed according to the parameter values reported in [86]. The other four schemes perform OCS, using oracles, synthetic data, a reference classifier, and meta-features to discard faulty subjects:

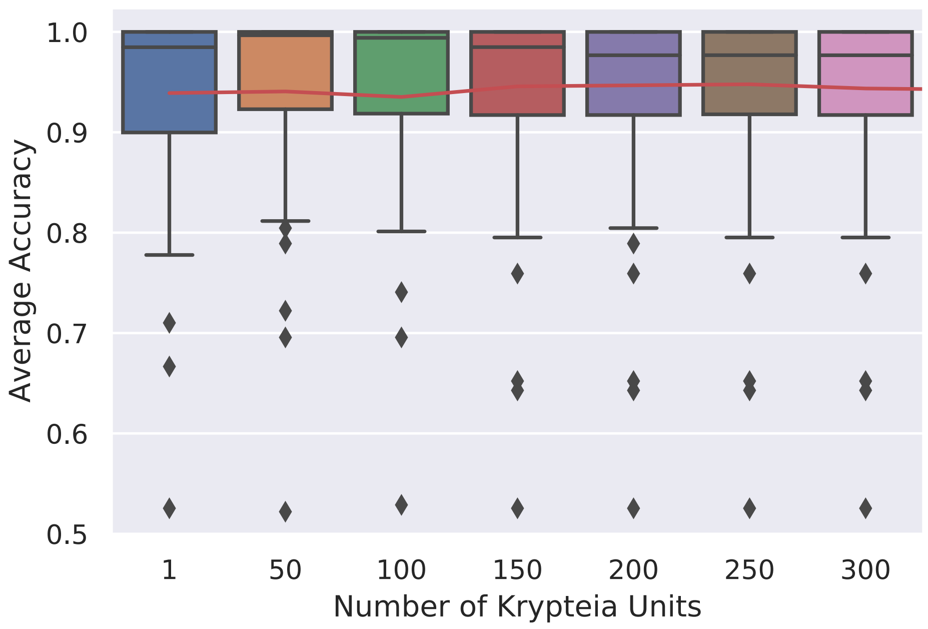

Fig. 3. Average accuracy for all the studied datasets using ensembles of n randomly chosen Krypteia Units in the Krypteia Army studied in Section 6.5.

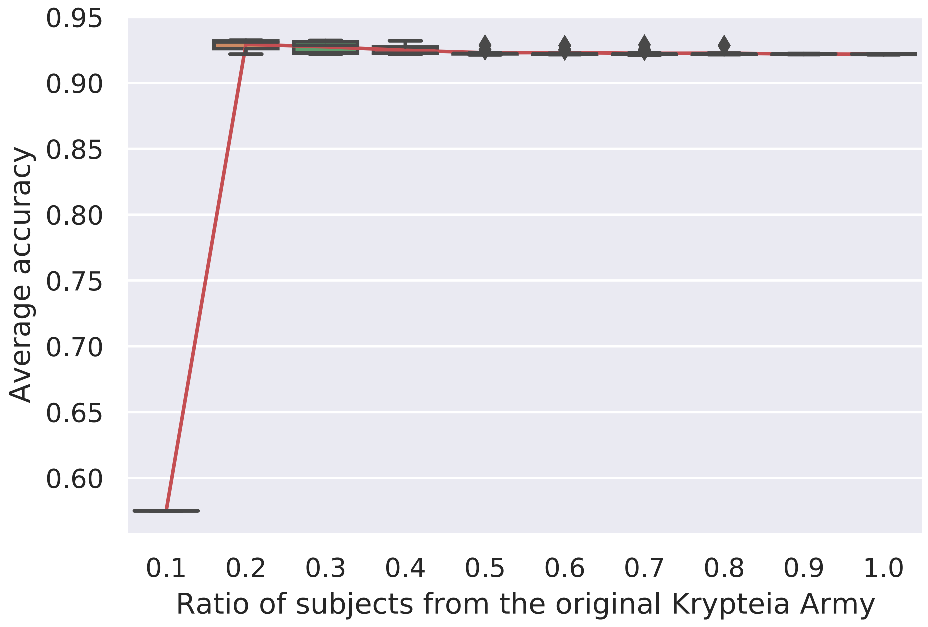

Fig. 4. Average accuracy for all the studied datasets using ensembles of n random weak learners from the original Krypteia Army studied in Section 6.5.

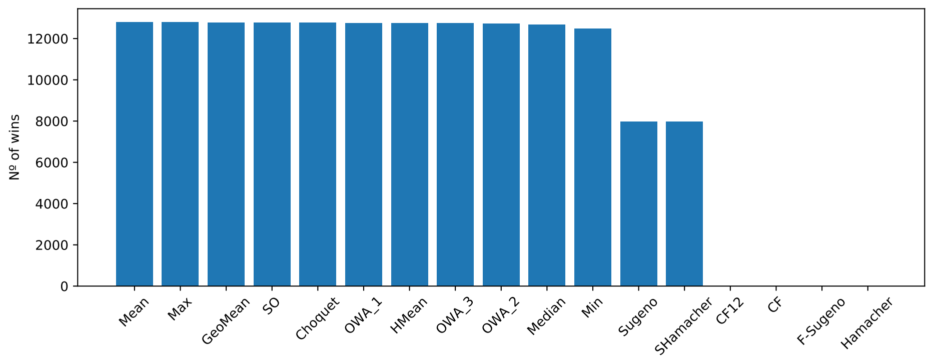

Fig. 5. Number of times each aggregation yielded the best results in all the experiments performed with Krypteia ensembles.

Table 6
Results for the Krypteia divisions.

Dataset	Average	Best division	Best result	Dataset	Average	Best division	Best result
abalone	100.00	100.00	100.00	nursery	89.97	99.96	99.96
appendicitis	85.71	85.71	85.71	optdigits	94.76	95.28	95.37
australian	81.30	81.88	81.88	page-blocks	38.88	97.26	97.26
balance	100.00	100.00	100.00	phoneme	89.64	89.64	89.82
banana	88.61	89.81	89.81	penbased	98.99	99.04	99.40
bands	100.00	100.00	100.00	pima	100.00	100.00	100.00
breast	100.00	100.00	100.00	post-operative	40.00	100.00	100.00
bupa	67.25	66.67	69.57	ring	91.86	90.88	92.57
car	100.00	100.00	100.00	saheart	100.00	100.00	100.00
chess	100.00	100.00	100.00	satimage	91.10	90.83	91.30
coil2000	92.04	92.16	92.47	segment	98.46	98.48	98.70
contraceptive	52.81	52.54	53.56	sonar	100.00	100.00	100.00
crx	100.00	100.00	100.00	spambase	92.74	92.72	93.26
dermatology	86.67	95.83	98.61	spectfheart	75.00	77.78	77.78
ecoli	97.01	97.01	97.01	splice	100.00	100.00	100.00
flare	100.00	100.00	100.00	texture	94.35	94.36	94.91
german	90.00	100.00	100.00	thyroid	99.82	99.86	99.86
haberman	100.00	100.00	100.00	tic-tac-toe	100.00	100.00	100.00
housevotes	100.00	100.00	100.00	titanic	72.41	80.45	80.45
ionosphere	100.00	100.00	100.00	twonorm	92.24	92.57	93.11
iris	100.00	100.00	100.00	vehicle	99.41	99.41	99.41
letter	69.81	99.72	100.00	vowel	90.81	90.91	91.92
magic	100.00	100.00	100.00	wdbc	100.00	100.00	100.00
mammographic	78.98	79.52	80.12	wine	100.00	100.00	100.00
mushroom	100.00	100.00	100.00	wisconsin	93.14	92.70	93.43
newthyroid	97.67	97.67	97.67	yeast	99.73	100.00	100.00

Table 7
Performance summary for the Krypteia divisions.

	Average division	Best division	Best result
Average Acc.	94.09	94.49	**94.58**

- Adaboost [21]: it serially trains each classifier. In each iteration, it weights each instance according to its difficulty to be classified, aiming to correctly classify it in the next iteration. For our experimentation, we have used 50 decision trees to form the Adaboost.
- Bagging (Bootstrap Aggregation) [20]: it aims to increase accuracy by combining the outputs of the classifiers in the ensemble that were trained using different subsamples of the original data. Sampling with replacement is used to train all the classifiers in the ensemble and thus some of the instances may appear more than once in the training set. For our experimentation, we have used 10 decision trees to form the Bagging classifier.
- Majority vote SVM: it consists of different SVM classifiers trained with a different kernel. For our case, we have trained 5 different RBF kernels classifiers with five different σ parameters evenly spaced such as: $[0.5, 1.5]/(number\,of\,features)$.
- Random Forest [87]: it combines the output of many different decision trees computed from different subsamples of the original data. The final decision is taken as the majority vote of all of the trees outputs'. We have set as 100 the number of trees in the Random Forest for our experimentation.
- K-Nearest Oracles Eliminate (K-NORAE) [88]: it selects the classifiers that correctly classify all the samples in their region of competence.
- Dynamic Ensemble Selection Multiclass Imbalance (DES-MI) [89]: it generates artificial training sets of randomly balanced data and then chooses the classifiers that correctly discriminated the minority class samples.
- Randomized Reference Classifier (DES-RRC) [90]: it combines Dynamic Ensemble Selection with a measure to evaluate each possible classifier in the final ensemble using a reference classifier.
- META-DES [53]: it selects a set of classifiers from a list, using five different meta-features to test each classifier's competence.
- Extreme Gradient Boosting (XGBoost) [91]: gradient boosting is a generalization of Adaboost that consist of using a differentiable loss function. This function is optimized using a gradient descent procedure, so that in each step a new weak learner is included to reduce the loss of the system. Gradient boosting is considered to be the state-of-the-art in classification of tabular data [92].

We computed the analogous experiments with these ensemble classifiers as in the Krypteia ensembles. Table 9 collects the average accuracy for all the datasets for the different ensemble classifiers. We found the Krypteia Army to be the most effective classifier, followed by the Best Krypteia Division and the Best Krypteia Unit. All of them outperform the best performing of the seven classifier ensembles tested, DES-MI. In Table 10 we have computed how many times each algorithm performed best (counting ties as winnings for both). We found that most of the times the Krypteia Army showed the best result and that every Krypteia variant again outperform all the benchmarking classifier ensembles.

In Table 11 we show the *P*-values obtained using a Wilcoxon statistical test, comparing all the ensembles used with the Krypteia Army, which was the ensemble with the highest average accuracy. We found statistical differences favouring the Krypteia Army compared to any of the other classifiers tested.

Finally, we have tested the performance of the Krypteia algorithm in a setting with higher dimensionality and more samples. In order to do so, we have used a dataset that is very popular in the deep learning literature: CIFAR10 [93]. This dataset consists of 600 000 images that belong to 10 different classes, 50 000 used for training and 10 000 for test. In order to apply our proposal to this dataset, we used a LeNet convolutional network architecture [94]. Once the network is trained, we discard the last layer of this network and compute its output for each sample. We use this output as the features to train the different classification algorithms.

In Table 12 we show the accuracy results in the test set for the Krypteia Army and two benchmarking classical ensemble approaches that performed better, Random Forest and Extreme Gradient boosting. We can see that both the Random Forest and the Extreme Gradient Boosting performed worse than the original performance of this LeNet architecture for this dataset, that achieved an accuracy rate of 65.84, while the Krypteia army performed slightly better (65.87) and performed the best overall. This is an interesting result keeping in mind

Table 8
Performance for the Krypteia army, with the corresponding aggregations that obtained that result.

Dataset	Ag1	Ag2	Ag3	Acc.	Dataset	Ag1	Ag2	Ag3	Acc.
abalone	Mean	Mean	Mean	100	nursery	Mean	Min	Min	100
appendicitis	Median	Min	Median	90.4	optdigits	Min	Max	S-Ham.	98.1
australian	Min	Max	S-Ham.	86.9	page-blocks	Choquet	Max	Min	97.5
balance	Mean	Mean	Mean	100	penbased	Min	Max	Min	99.0
banana	Min	Mean	Max	90.5	phoneme	Max	Median	Sugeno	90.6
bands	Mean	Mean	Mean	100	pima	Mean	Mean	Mean	100
breast	Mean	Mean	Mean	100	post-operative	Mean	Mean	Mean	100
bupa	Min	Mean	Max	72.4	ring	Min	Max	S-Ham.	97.7
car	Mean	Mean	Mean	100	saheart	Mean	Mean	Mean	100
chess	Mean	Mean	Mean	100	satimage	GM	Max	Median	91.9
coil2000	OWA_1	Max	Max	94.5	segment	S-Ham.	Min	OWA_3	99.1
contraceptive	Min	Max	Min	55.9	sonar	Mean	Mean	Mean	100
crx	Mean	Mean	Mean	100	spambase	OWA_2	Min	OWA_2	93.8
dermatology	Mean	Min	Max	98.6	spectfheart	HM	Min	Median	81.4
ecoli	Mean	Min	Max	100	splice	Mean	Mean	Mean	100
flare	Mean	Mean	Mean	100	texture	HM	Max	Max	98.5
german	Mean	Mean	Mean	100	thyroid	Mean	Mean	Mean	99.8
haberman	Mean	Mean	Mean	100	tic-tac-toe	Mean	Mean	Mean	100
housevotes	Mean	Mean	Mean	100	titanic	Mean	Mean	Mean	80.4
ionosphere	Mean	Mean	Mean	100	twonorm	Max	Choquet	OWA_1	97.7
iris	Mean	Mean	Mean	100	vehicle	Mean	Min	Mean	100
letter	OWA_3	Min	OWA_3	100	vowel	Min	Max	Mean	94.9
magic	Mean	Mean	Mean	100	wdbc	Mean	Mean	Mean	100
mammographic	Min	Max	Sugeno	83.1	wine	Mean	Mean	Mean	100
mushroom	Mean	Mean	Mean	100	wisconsin	Min	Mean	Max	94.8
newthyroid	Min	Max	Min	100	yeast	Min	Median	Median	100

Table 9
Average performance for different ensemble classifiers and the best instance of Krypteia ensemble classifiers used.

Algorithm	Accuracy
Adaboost	83.84 ± 19.65
Bagging	90.99 ± 9.81
Majority vote SVM	68.90 ± 21.84
Random forest	92.61 ± 8.53
K-NORAE	90.05 ± 11.00
DES-MI	93.87 ± 10.59
DES-RRC	92.74 ± 10.72
META-DES	93.60 ± 10.70
XGBoost	95.14 ± 9.81
SVM-Krypteia mean	93.30 ± 10.34
Krypteia Unit	94.09 ± 14.36
Krypteia division	94.49 ± 12.85
Krypteia army	$\mathbf{95.83 \pm 8.35}$

Table 10
Number of times each ensemble beat the others in the different datasets. Ties are considered as wins for both.

Algorithm	Nº of times that won
Adaboost	5
Bagging	24
Majority vote SVM	24
Random forest	25
K-NORAE	23
DES-MI	22
DES-RRC	22
META-DES	21
XGBoost	**26**
Krypteia Unit	25
Krypteia division	**35**
Krypteia army	**39**

Table 11
P-values for all the ensembles compared to the Krypteia army.

Algorithm	*P*-value
Adaboost	$P < .001$
Bagging	$P < .001$
Majority vote SVM	$P < .001$
Random forest	$P < .001$
K-NORAE	$P < .001$
DES-MI	$P < .001$
DES-RRC	$P < .001$
META-DES	$P < .001$
XGBoost	$P < .001$
Krypteia Unit	$P < .001$
Krypteia division	$P < .001$

that the Krypteia approach was not initially designed to handle such high dimensional classification problems. In fact, we aim to extend our approach to deal with these kinds of problems in the short future.

8. Discussion of the empirical results obtained

In the results presented in this the Krypteia ensemble obtained favourable results compared to the rest of the ensemble classifiers tested. The best proposal, the Krypteia Army, also presented the lowest standard deviation compared to the rest of its competitors and Krypteia schemes.

The SVM-only mean-only Krypteia performed well, but not better than the XGBoost, DES-MI, and META-DES approaches. This might help explain why the other Krypteia schemes performed better: adding more variability in the aggregation process and the classifiers did significantly improve the performance of the system. The CART analysis collected in Table 5 showed that the unit size and specially the survival threshold were instrumental for a unit performance, which can also explain the advantages in performance with respect to the models that used OCS based on their individual accuracy: the Krypteia not only discards subjects, but keeps generating more that are potentially useful. Those that are redundant can be then discarded by the VGs in the social ordeal.

One of the main advantages of the Krypteia scheme over its competitors, is that it does not really have any hyperparameter to be tuned,

Table 12
Accuracy values for the CIFAR10 dataset. A LeNet model is considered to compute the features for each image.

Algorithm	Accuracy
LeNet base performance	65.84
Extreme Gradient boosting	64.09
Random forest	65.37
Krypteia army	65.87

as almost every parameter is sampled from a random distribution. The only parameters that cannot be stochastically fixed are the number of units in the Krypteia Army and the aggregation functions used, that we chose using a five-fold validation on the training set. On the contrary, it has quite different steps, which makes its computation more complex than its rivals. The CART analysis performed in this study could be a good starting point to choose those elements that could be simplified without affecting performance.

Although there is not a theoretical limit for the size of the problems the Krypteia scheme can tackle, this scheme is designed for data that can fit completely in memory. When using other paradigms for datasets of bigger sizes, we must take into account further consideration to fully support and exploit mini-batches or map-reduce approaches. For example, it is possible to use classifiers that perform some kind of online-learning, or it is also possible to use dedicated classifiers for each batch. Besides, the use of VGs can be focused in keeping the classifiers that performed best on the data that is significantly different to what the system has already processed. The performance of all these options needs to be tested properly in order to determine which is the best option to scale the Krypteia scheme to large databases and will be a subject of study in future works.

9. Conclusions and future lines

In this paper we presented the Krypteia ensemble: a new form to generate classifier ensembles based on an ancient Spartan ritual to train the future elites of their society. We detailed the process needed to compute these classifiers and we explained how they relate to this ancient tradition, by exposing each different subject to distinct hardships. Then, we studied how different aggregation functions work in different Krypteia ensembles.

We tested the different Krypteia ensembles on a large experimental study including 52 datasets. We have studied the performance of different forms of the Krypteia ensemble, and the effect of the various parameters that define the Krypteia training process. Then, we compared the results obtained by the Krypteia against 7 other ensemble design algorithms, obtaining significantly better results.

Future research shall aim to improve the way in which Krypteia Units are assembled to form Krypteia Divisions. We are also interested in expanding the Krypteia scheme to support environments where the data size is too big to fit in memory, like in deep learning, where the dataset is loaded in mini-batches, and big data. Refining the performance of the different ordeals, i.e. adding label perturbation and different loss functions to the survival ordeal, will also be explored. Finally, we also intend to study the application of the Krypteia and other social phenomena to other domains different from classification [95].

CRediT authorship contribution statement

J. Fumanal-Idocin: Conceptualization, Methodology, Software, Writing – original draft. **O. Cordón:** Writing – review & editing. **H. Bustince:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

Javier Fumanal Idocin and Humberto Bustince's research has been supported by project PID2019-108392 GB I00 (AEI/10.13039/501100011033).

Oscar Cordón's research has been funded by the Spanish Ministry of Science and Innovation (MICIN), Agencia Estatal de Investigación (AEI), Spain, under grant CONFIA (PID2021-122916NB-I00), and by the Regional Government of Andalusia under grant EXAISFI (P18-FR-4262), both including European Regional Development Funds (ERDF).

Open access funding provided by Universidad Pública de Navarra.

References

[1] M. Berthold, D.J. Hand, Intelligent Data Analysis, Springer, 2003.
[2] S. Brandt, Data Analysis, Springer, 1998.
[3] D.A. Keim, Information visualization and visual data mining, IEEE Trans. Vis. Comput. Graphics 8 (1) (2002) 1–8.
[4] R. Cooley, B. Mobasher, J. Srivastava, Web mining: Information and pattern discovery on the world wide web, in: Proceedings Ninth IEEE International Conference on Tools with Artificial Intelligence, IEEE, 1997, pp. 558–567.
[5] N. Zhong, Y. Li, S.-T. Wu, Effective pattern discovery for text mining, IEEE Trans. Knowl. Data Eng. 24 (1) (2010) 30–44.
[6] J.-P. Brunet, P. Tamayo, T.R. Golub, J.P. Mesirov, Metagenes and molecular pattern discovery using matrix factorization, Proc. Natl. Acad. Sci. 101 (12) (2004) 4164–4169.
[7] M.M. Petrou, C. Petrou, Image Processing: The Fundamentals, John Wiley & Sons, 2010.
[8] C.C. Aggarwal, Data classification, in: Data Mining, Springer, 2015, pp. 285–344.
[9] H. Ramchoun, M.A.J. Idrissi, Y. Ghanou, M. Ettaouil, Multilayer perceptron: Architecture optimization and training, Int. J. Interact. Multimedia Artif. Intell. 4 (1) (2016) 26–30.
[10] A.F.M. Hani, H.A. Nugroho, H. Nugroho, Gaussian Bayes classifier for medical diagnosis and grading: application to diabetic retinopathy, in: 2010 IEEE EMBS Conference on Biomedical Engineering and Sciences, IECBES, IEEE, 2010, pp. 52–56.
[11] D. Ververidis, C. Kotropoulos, Emotional speech classification using Gaussian mixture models, in: 2005 IEEE International Symposium on Circuits and Systems, IEEE, 2005, pp. 2871–2874.
[12] L.E. Peterson, K-nearest neighbor, Scholarpedia 4 (2) (2009) 1883.
[13] C. Cortes, V. Vapnik, Support vector machine, Mach. Learn. 20 (3) (1995) 273–297.
[14] N. Saleena, et al., An ensemble classification system for twitter sentiment analysis, Procedia Comput. Sci. 132 (2018) 937–946.
[15] X. Feng, Z. Xiao, B. Zhong, J. Qiu, Y. Dong, Dynamic ensemble classification for credit scoring using soft probability, Appl. Soft Comput. 65 (2018) 139–151.
[16] C.B.C. Latha, S.C. Jeeva, Improving the accuracy of prediction of heart disease risk based on ensemble classification techniques, Inform. Med. Unlocked 16 (2019) 100203.
[17] L. Rokach, Ensemble-based classifiers, Artif. Intell. Rev. 33 (1–2) (2010) 1–39.
[18] R. Polikar, Ensemble based systems in decision making, IEEE Circuits Syst. Mag. 6 (3) (2006) 21–45.
[19] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[20] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[21] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1) (1997) 119–139.
[22] A. Rojarath, W. Songpan, C. Pong-inwong, Improved ensemble learning for classification techniques based on majority voting, in: 2016 7th IEEE International Conference on Software Engineering and Service Science, ICSESS, IEEE, 2016, pp. 107–110.
[23] R.M. Cruz, D.V. Oliveira, G.D. Cavalcanti, R. Sabourin, FIRE-DES++: Enhanced online pruning of base classifiers for dynamic ensemble selection, Pattern Recognit. 85 (2019) 149–160.
[24] A. Onan, S. Korukoğlu, H. Bulut, A multiobjective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification, Expert Syst. Appl. 62 (2016) 1–16.

[25] J.M. Moyano, E.L. Gibaja, K.J. Cios, S. Ventura, Review of ensembles of multi-label classifiers: models, experimental study and prospects, Inf. Fusion 44 (2018) 33–45.
[26] D. Partridge, W.B. Yates, Engineering multiversion neural-net systems, Neural Comput. 8 (4) (1996) 869–893.
[27] L. Wang, T. Mo, X. Wang, W. Chen, Q. He, X. Li, S. Zhang, R. Yang, J. Wu, X. Gu, et al., A hierarchical fusion framework to integrate homogeneous and heterogeneous classifiers for medical decision-making, Knowl.-Based Syst. 212 (2021) 106517.
[28] G. Beliakov, H. Bustince, T. Calvo, A Practical Guide to Averaging Functions, Vol. 329, Springer, 2016.
[29] G.P. Dimuro, J. Fernández, B. Bedregal, R. Mesiar, J.A. Sanz, G. Lucca, H. Bustince, The state-of-art of the generalizations of the Choquet integral: from aggregation and pre-aggregation to ordered directionally monotone functions, Inf. Fusion 57 (2020) 27–43.
[30] S. Abbaszadeh, E. Hullermeier, Machine learning with the Sugeno Integral: The case of binary classification, IEEE Trans. Fuzzy Syst. (2020).
[31] L. De Miguel, M. Sesma-Sara, M. Elkano, M. Asiain, H. Bustince, An algorithm for group decision making using n-dimensional fuzzy sets, admissible orders and OWA operators, Inf. Fusion 37 (2017) 126–131.
[32] L. De Miguel, D. Gómez, J.T. Rodríguez, J. Montero, H. Bustince, G.P. Dimuro, J.A. Sanz, General overlap functions, Fuzzy Sets and Systems 372 (2019) 81–96.
[33] H. Bustince, G. Beliakov, G.P. Dimuro, B. Bedregal, R. Mesiar, On the definition of penalty functions in data aggregation, Fuzzy Sets and Systems 323 (2017) 1–18.
[34] S.T. Hadjitodorov, L.I. Kuncheva, L.P. Todorova, Moderate diversity for better cluster ensembles, Inf. Fusion 7 (3) (2006) 264–275.
[35] V. Bolón-Canedo, A. Alonso-Betanzos, Ensembles for feature selection: A review and future trends, Inf. Fusion 52 (2019) 1–12.
[36] A.J. Bayliss, The Spartans, Oxford University Press, 2020.
[37] W. Wang, Some fundamental issues in ensemble methods, in: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 2243–2250.
[38] L.I. Kuncheva, C.J. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, Mach. Learn. 51 (2) (2003) 181–207.
[39] Q. Dai, R. Ye, Z. Liu, Considering diversity and accuracy simultaneously for ensemble pruning, Appl. Soft Comput. 58 (2017) 75–91.
[40] L. Liu, W. Wei, K.-H. Chow, M. Loper, E. Gursoy, S. Truex, Y. Wu, Deep neural network ensembles against deception: Ensemble diversity, accuracy and robustness, in: 2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems, MASS, IEEE, 2019, pp. 274–282.
[41] A. Rahman, S. Tasnim, Ensemble classifiers and their applications: A review, 2014, arXiv preprint arXiv:1404.4088.
[42] H. Drucker, C. Cortes, L.D. Jackel, Y. LeCun, V. Vapnik, Boosting and other ensemble methods, Neural Comput. 6 (6) (1994) 1289–1301.
[43] L. Rokach, O. Maimon, I. Lavi, Space decomposition in data mining: A clustering approach, in: International Symposium on Methodologies for Intelligent Systems, Springer, 2003, pp. 24–31.
[44] B. Seijo-Pardo, I. Porto-Díaz, V. Bolón-Canedo, A. Alonso-Betanzos, Ensemble feature selection: homogeneous and heterogeneous approaches, Knowl.-Based Syst. 118 (2017) 124–139.
[45] Z.H. Kilimci, S. Akyokus, Deep learning-and word embedding-based heterogeneous classifier ensembles for text classification, Complexity 2018 (2018).
[46] J.N. van Rijn, G. Holmes, B. Pfahringer, J. Vanschoren, The online performance estimation framework: heterogeneous ensemble learning for data streams, Mach. Learn. 107 (1) (2018) 149–176.
[47] R. Lysiak, M. Kurzynski, T. Woloszynski, Optimal selection of ensemble classifiers using measures of competence and diversity of base classifiers, Neurocomputing 126 (2014) 29–35, Recent trends in Intelligent Data Analysis Online Data Processing.
[48] H.G. Zefrehi, H. Altınçay, Imbalance learning using heterogeneous ensembles, Expert Syst. Appl. 142 (2020) 113005.
[49] R.M. Cruz, R. Sabourin, G.D. Cavalcanti, Dynamic classifier selection: Recent advances and perspectives, Inf. Fusion 41 (2018) 195–216.
[50] V.Y. Kulkarni, P.K. Sinha, Pruning of random forest classifiers: A survey and future directions, in: 2012 International Conference on Data Science & Engineering, ICDSE, IEEE, 2012, pp. 64–68.
[51] R. Hu, S. Zhou, Y. Liu, Z. Tang, Margin-based Pareto ensemble pruning: An ensemble pruning algorithm that learns to search optimized ensembles, Comput. Intell. Neurosci. 2019 (2019).
[52] C. Lin, W. Chen, C. Qiu, Y. Wu, S. Krishnan, Q. Zou, LibD3C: ensemble classifiers with a clustering and dynamic selection strategy, Neurocomputing 123 (2014) 424–435.
[53] R.M. Cruz, R. Sabourin, G.D. Cavalcanti, T.I. Ren, META-DES: A dynamic ensemble selection framework using meta-learning, Pattern Recognit. 48 (5) (2015) 1925–1935.
[54] K. Trawiński, O. Cordón, A. Quirin, A study on the use of multiobjective genetic algorithms for classifier selection in FURIA-based fuzzy multiclassifiers, Int. J. Comput. Intell. Syst. 5 (2) (2012) 231–253.
[55] K. Trawinski, O. Cordón, A. Quirin, Embedding evolutionary multiobjective optimization into fuzzy linguistic combination method for fuzzy rule-based classifier ensembles, in: 2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE, 2014, pp. 1968–1975.
[56] R.C.T. de Souza, C.A. de Macedo, L. dos Santos Coelho, J. Pierezan, V.C. Mariani, Binary coyote optimization algorithm for feature selection, Pattern Recognit. 107 (2020) 107470.
[57] M. Elkano, M. Galar, J.A. Sanz, P.F. Schiavo, S. Pereira Jr., G.P. Dimuro, E.N. Borges, H. Bustince, Consensus via penalty functions for decision making in ensembles in fuzzy rule-based classification systems, Appl. Soft Comput. 67 (2018) 728–740.
[58] L. Rokach, Pattern Classification using Ensemble Methods, Vol. 75, World Scientific, 2010.
[59] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, F. Herrera, Ordering-based pruning for improving the performance of ensembles of classifiers in the framework of imbalanced datasets, Inform. Sci. 354 (2016) 178–196.
[60] J. Fumanal-Idocin, Y.-K. Wang, C.-T. Lin, J. Fernández, J.A. Sanz, H. Bustince, Motor-imagery-based brain-computer interface using signal derivation and aggregation functions, IEEE Trans. Cybern. (2021) 1–12, http://dx.doi.org/10.1109/TCYB.2021.3073210.
[61] G. Lucca, J.A. Sanz, G.P. Dimuro, B. Bedregal, H. Bustince, R. Mesiar, CF-integrals: A new family of pre-aggregation functions with application to fuzzy rule-based classification systems, Inform. Sci. 435 (2018) 94–110.
[62] F. Bardozzo, B. De La Osa, L. Horanská, J. Fumanal-Idocin, M. delli Priscoli, L. Troiano, R. Tagliaferri, J. Fernandez, H. Bustince, Sugeno integral generalization applied to improve adaptive image binarization, Inf. Fusion 68 (2021) 37–45.
[63] G. Beliakov, J.-Z. Wu, Learning fuzzy measures from data: simplifications and optimisation strategies, Inform. Sci. 494 (2019) 100–113.
[64] K. Trawiński, O. Cordón, L. Sánchez, A. Quirin, A genetic fuzzy linguistic combination method for fuzzy rule-based multiclassifiers, IEEE Trans. Fuzzy Syst. 21 (5) (2013) 950–965.
[65] M. Papčo, I. Rodríguez-Martínez, J. Fumanal-Idocin, A.H. Altalhi, H. Bustince, A fusion method for multi-valued data, Inf. Fusion 71 (2021) 1–10.
[66] M. Gupta, J. Qi, Theory of T-norms and fuzzy inference methods, Fuzzy Sets and Systems 40 (3) (1991) 431–450.
[67] R.R. Yager, J. Kacprzyk, The Ordered Weighted Averaging Operators: Theory and Applications, Springer Science & Business Media, 2012.
[68] G.P. Dimuro, G. Lucca, B. Bedregal, R. Mesiar, J.A. Sanz, C.-T. Lin, H. Bustince, Generalized cf1f2-integrals: From Choquet-like aggregation to ordered directionally monotone functions, Fuzzy Sets and Systems 378 (2020) 44–67.
[69] H. Bustince, R. Mesiar, J. Fernandez, M. Galar, D. Paternain, A. Altalhi, G. Dimuro, B. Bedregal, Z. Takáč, D-Choquet integrals: Choquet integrals based on dissimilarities, Fuzzy Sets and Systems 414 (2021) 1–27, Aggregation Functions.
[70] G. Lucca, J. Antonio Sanz, G. Pereira Dimuro, B. Bedregal, R. Mesiar, A. Kolesarova, H. Bustince, Preaggregation functions: Construction and an application, IEEE Trans. Fuzzy Syst. 24 (2) (2016) 260–272.
[71] M. Sugeno, Theory of Fuzzy Integrals and its Applications (Ph.D. thesis), Tokyo Institute of Technology, 1974.
[72] L.-W. Ko, Y.-C. Lu, H. Bustince, Y.-C. Chang, Y. Chang, J. Ferandez, Y.-K. Wang, J.A. Sanz, G.P. Dimuro, C.-T. Lin, Multimodal fuzzy fusion for enhancing the motor-imagery-based brain computer interface, IEEE Comput. Intell. Mag. 14 (1) (2019) 96–106.
[73] D. Futter, Plutarch, Plato and Sparta, Akroterion 57 (1) (2012) 35–51.
[74] J. Ducat, Spartan Education: Youth and Society in the Classical Period, ISD LLC, 2006.
[75] I. Joseph, The deadliest blogger: Military history page, 2021, https://deadliestblogpage.wordpress.com/. Accessed: 2021-04-26.
[76] A.J. Izenman, Linear discriminant analysis, in: Modern Multivariate Statistical Techniques, Springer, 2013, pp. 237–280.
[77] S. Srivastava, M.R. Gupta, B.A. Frigyik, Bayesian quadratic discriminant analysis, J. Mach. Learn. Res. 8 (Jun) (2007) 1277–1305.
[78] Y. Freund, L. Mason, The alternating decision tree learning algorithm, in: International Conference on Machine Learning, Vol. 99, 1999, pp. 124–133.
[79] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artificial Intelligence Res. 16 (2002) 321–357.
[80] G.E. Batista, R.C. Prati, M.C. Monard, A study of the behavior of several methods for balancing machine learning training data, ACM SIGKDD Explor. Newsl. 6 (1) (2004) 20–29.
[81] D.L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Trans. Syst. Man Cybern. SMC-2 (3) (1972) 408–421.
[82] X.-S. Yang, S. Deb, S. Fong, Metaheuristic algorithms: optimal balance of intensification and diversification, Appl. Math. Inf. Sci. 8 (3) (2014) 977.
[83] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, J. Mult.-Valued Logic Soft Comput. 17 (2011).
[84] E. Bradford, Thermopylae: The Battle for the West, Open Road Media, 2014.
[85] E.A. Antipov, E.B. Pokryshevskaya, Mass appraisal of residential apartments: An application of random forest for valuation and a CART-based approach for model diagnostics, Expert Syst. Appl. 39 (2) (2012) 1772–1778.

[86] U. Agrawal, A.J. Pinar, C. Wagner, T.C. Havens, D. Soria, J.M. Garibaldi, Comparison of fuzzy integral-fuzzy measure based ensemble algorithms with the state-of-the-art ensemble algorithms, in: International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Springer, 2018, pp. 329–341.
[87] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[88] A.H. Ko, R. Sabourin, A.S. Britto Jr., From dynamic classifier selection to dynamic ensemble selection, Pattern Recognit. 41 (5) (2008) 1718–1731.
[89] S. García, Z.-L. Zhang, A. Altalhi, S. Alshomrani, F. Herrera, Dynamic ensemble selection for multi-class imbalanced datasets, Inform. Sci. 445 (2018) 22–37.
[90] T. Woloszynski, M. Kurzynski, A probabilistic model of classifier competence for dynamic ensemble selection, Pattern Recognit. 44 (10–11) (2011) 2656–2668.
[91] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.
[92] C. Bentéjac, A. Csörgő, G. Martínez-Muñoz, A comparative analysis of gradient boosting algorithms, Artif. Intell. Rev. 54 (3) (2021) 1937–1967.
[93] A. Krizhevsky, G. Hinton, Learning Multiple Layers of Features from Tiny Images, Technical Report, Computer Science Department, University of Toronto, 2009.
[94] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[95] J. Fumanal-Idocin, A. Alonso-Betanzos, Ó. Cordón, H. Bustince, M. Minárová, Community detection and social network analysis based on the Italian wars of the 15th century, Future Gener. Comput. Syst. 113 (2020) 25–40.