The Knowledge Engineering Review, Vol. 38:e1, 1–41. © 2023, Cambridge University Press
DOI: 10.1017/S0269888923000024 Printed in the United Kingdom

A Survey of Evolutionary Algorithms for Supervised Ensemble Learning

Henry E.L. Cagnini[1], Silvia C.N. das Dôres[1], Alex A. Freitas[2], Rodrigo C. Barros[1]

[1] *School of Technology, PUCRS, Avenida Ipiranga 6681, Porto Alegre, RS, 90619-900, Brazil*
E-mail: henry.cagnini@edu.pucrs.br, silvia.dores@acad.pucrs.br, rodrigo.barros@pucrs.br
[2] *Computing School, University of Kent, Giles Ln, Canterbury, CT2 7NZ, UK*
E-mail: a.a.freitas@kent.ac.uk

Abstract

This paper presents a comprehensive review of evolutionary algorithms that learn an ensemble of predictive models for supervised machine learning (classification and regression). We propose a detailed four-level taxonomy of studies in this area. The first level of the taxonomy categorizes studies based on which stage of the ensemble learning process is addressed by the evolutionary algorithm: the generation of base models, model selection, or the integration of outputs. The next three levels of the taxonomy further categorize studies based on methods used to address each stage. In addition, we categorize studies according to the main types of objectives optimized by the evolutionary algorithm, the type of base learner used and the type of evolutionary algorithm used. We also discuss controversial topics, like the pros and cons of the selection stage of ensemble learning, and the need for using a diversity measure for the ensemble's members in the fitness function. Finally, as conclusions, we summarize our findings about patterns in the frequency of use of different methods, and suggest several new research directions for evolutionary ensemble learning.

1 Introduction

Supervised ensemble learning – sometimes referred to as a mixture of experts, classifier ensembles, or multiple classifier system [170, 137] is a paradigm within the machine learning area concerned with integrating multiple base supervised learners in order to produce better predictive models than simply learning a single strong model. An ensemble typically performs its predictions by using a voting mechanism (e.g. majority voting) that computes the mean or the mode of the predictions output by the ensemble's members (base learners). Ensemble learning methods have won several academic and industrial machine learning competitions [168], and such methods have been extensively deployed in real-world AI applications [184, 149].

Ensembles have several advantages over a single learner: (i) it is usually computationally cheaper to integrate a set of simple, weak models than to induce a single robust, complex model [114]; (ii) ensembles composed by classifiers that are, in turn, only slightly better than random guessing, can still present predictive performance comparable to a strong single classifier [74, 90, 126]; (iii) different base learners can be specialized in different regions of the input space, making their consensus more flexible and effective when dealing with complex problems [74]. Indeed, there is both theoretical and empirical evidence demonstrating that a good ensemble can be obtained by combining individual models that make distinct errors (e.g., errors on different parts of the input space) [98, 82, 147, 115, 84].

Ensemble learning comprises three distinct stages, whose names vary in the literature: generation, selection, and integration [20, 123, 25], which is the most common naming system,

and the one that will be used in this paper; pre-gate, ensemble-member, and post-gate [51]; or generation, pruning, and fusion [151]. The three-step generation of ensembles can be reduced to a hypothesis-search problem in combinatorial spaces, and so it is often approached by a variety of heuristic approaches, such as boosting [74], bagging [18], and Evolutionary Algorithms (EAs) [123, 120].

EAs have several advantages for ensemble learning, such as: performing a global search, which is less likely to get trapped into local optima than greedy search methods [207]; being easily adapted for multi-objective optimization [48]; and dealing with multiple solutions in parallel, due to their population-based nature, e.g. [85, 117]. Hence, many EAs for supervised ensemble learning have been proposed in the literature in the past few years.

Several application domains have benefited from EA-based ensemble learning algorithms, including e.g. wind speed prediction [213, 205], cancer detection [182, 112, 111, 108, 109, 107], noise by-pass detection in vehicles [163], stock market prediction [30, 132, 133], and microarray data classification [127, 124, 154, 153, 125, 97, 98, 29, 100, 169, 4, 162], to name just a few.

This paper presents a survey of EAs for supervised ensemble learning. Our main contribution is to properly identify, categorize, and evaluate the available research studies in this area. This survey is aimed towards researchers on evolutionary algorithms and/or ensemble learning algorithms.

As related work, several surveys have been published on ensemble studies from different perspectives. Regarding specifically EAs for supervised ensemble learning, Yao and Islam [210] present a review of EAs for designing ensembles, but they focus only on artificial neural networks as the base learners to be combined. Sagi and Rokach [168], as well as Dietterich [54] present a general review of ensemble learning studies, based on traditional non-evolutionary methods. Rokach [165], Kotsiantis [106], and Tabassum and Ahmed [184] review ensembles designed only for classification tasks. Similarly, Mendes-Moreira et al. [136] and Vega-Pons et al. [195] review ensemble methods for regression and clustering tasks only, respectively. There are also papers on specific domain applications of ensembles, e.g. Athar et al. [8], which reviews ensemble methods for sentiment analysis; Gomes et al. [80] and Krawczyk et al. [113] also review ensemble learning for data stream classification and regression.

Despite the relevant contributions of the previously cited literature, this work is to the best of our knowledge the first review to focus on general-application EAs for supervised ensemble learning in a comprehensive fashion. In particular, we highlight the following contributions: i) we provide a general overview of EAs for supervised ensemble learning, not exclusively focusing on any specific EA or any given type of supervised model, but presenting an in-depth analysis of the different algorithms proposed for each stage of ensemble learning, with their respective advantages and pitfalls; and ii) we provide a detailed taxonomy to properly categorize supervised evolutionary ensembles, helping the reader to filter the literature and understand the possibilities when designing EAs for this task. Note that reviewing EAs for ensemble learning in unsupervised settings (e.g., the clustering task) is out of the scope of this paper.

The rest of this paper is organized as follows. Section 2 briefly reviews the most well-known types of ensemble learning methods. Section 3 presents our novel taxonomy to categorize EAs designed for supervised ensemble learning. Sections 4, 5, and 6 review the EAs employed for the three stages of ensemble learning: generation, selection, and integration. In the next sections, we focus on broader aspects of EAs for ensemble learning that are not specific to any single stage, as follows. Section 7 details types of fitness functions often used by EAs for ensemble learning. Section 8 summarizes the types of EAs used for ensemble learning. Section 9 points to the most common base learners within the EAs reviewed in this survey. Section 10 describes the complexity of evolutionary algorithms when applied to ensemble learning. Finally, in Section 11 we summarize our findings by identifying patterns in the frequency of use of different methods across the surveyed EAs, and identify future trends and interesting research directions in this area.

2 Ensemble Learning

There are three main motivations to combine multiple learners [54]: representational, statistical, and computational. The representational motivation is that combining multiple base learners may provide better predictive performance than a single strong learner. For example, the generalization ability of a neural network can be improved by using it as base learner within an ensemble [27]. In theory, no base learner will have the best predictive performance for all problems, as stated by the *No Free Lunch* theorem [204]; and in practice, selecting the best learner for any given dataset is a very difficult problem [65],[105], which can be addressed by integrating several good learners into an ensemble.

The statistical motivation is to avoid poor-performance by averaging the outputs of many base learners. While averaging the output of multiple base learners may not produce the overall best output, it is also unlikely that it will produce the worst possible output [86]. This is particularly the case for data with few data points, so overfitting is more likely.

Finally, the computational motivation is that some algorithms require several runs with distinct initializations in order to avoid falling into bad local minima. Gradient descent, for example, often requires several runs and further evaluation on a validation set in order to avoid being trapped into local minima. Thus, it seems reasonable to integrate these already-trained intermediate models into an ensemble, stabilizing and improving the system's overall performance [43, 192].

Ensemble learning became popular during the 1990's [43], with some of the most important work arising around that time: stacking in 1992 [203]; boosting in 1995 and 1996 [74, 75, 173]; bagging in 1996 [18]; and random forests in the early 2000's [19]. We call these methods *traditional* to differentiate them from EA-based ensembles, though they are also referred to as *preprocessing-based ensemble methods* in the EA literature [112]. We briefly review them next.

2.1 Boosting

Boosting refers to the technique of continuously enhancing the predictive performance of a weak learner [112]. We present here the popular AdaBoost algorithm, proposed by Freund and Schapire [74]. Given a set of predictive attributes $\mathbf{X}$ and a set of class labels Y, $y \in Y = \{-1, +1\}$, in its first iteration AdaBoost assigns equal importance (weights) to each instance in the training set, $D_1(i) = 1/N,\ i = 1, \ldots, N$, with N as the number of instances. For a given number of iterations G, AdaBoost trains a weak classifier based on the distribution D_g, and then computes its error $\epsilon_g = \mathrm{P}_{i \sim D_g}[h_g(x_i) \neq y_i]$. Instances that are harder to classify will have their weights increased, so it becomes more rewarding to the model to classify them correctly.

Candidate algorithms for boosting must support assignment of weights for instances. If this is not possible, a set of instances can be sampled from D_g and supplied to the g^{th} learner. Although boosting usually improves the predictive performance of a weak classifier, its performance suffers when faced with noisy instances, since failing to correctly classify those instances will iteratively improve their importance and hence lead the learner to overfitting [120].

2.2 Bagging

Bootstrap **agg**regat**ing**, or simply *bagging*, aims at reducing training instability when a learner is faced with a given data distribution [18]. It consists of generating B subsets of size N from the original training distribution $D(i) = 1/N$, $i = 1, \ldots, N$ with replacement, causing some instances to be present in more than one subset. As a result, some base learners have a tendency to favor such instances, having more opportunities to correctly predict their values. The predictions of all trained B learners are combined by computing their mean (regression task) or mode (classification task).

By sampling different subsets of instances for different classifiers, bagging implicitly injects diversity within the ensemble [120], whereas boosting explicitly does this by weighting the data distribution to focus the base learners' attention into more difficult instances [120, 81].

2.3 Stacking

Compared to bagging and boosting, *stacked generalization* or simply *stacking* [203] is a more flexible strategy for ensemble learning. The user can choose one or more types of base learners to be used in the ensemble (e.g. using only decision trees, or mixing them with artificial neural networks [175]). Then, each base learner will output a prediction, and all learners' predictions will be combined by a meta-learner (which can also be chosen) to produce a single output. Popular traditional meta-learners for stacking include linear regression (for regression) and logistic regression (for classification). In Section 6.1 we review evolutionary algorithms used for learning logistic regression, and linear regression algorithms' weights. Stacking often improves the overall predictive performance of ensembles, making it a popular method [137, 175].

2.4 Random Forests

Random forests, proposed by Breiman [19], is a type of ensemble learning method where both the base learner and data sampling are pre-determined: decision trees and random sampling of both instances and attributes. The training process for the original random forests algorithm [19] is described as follows. First, the algorithm randomly samples with replacement B subsets of training instances, one for building each of B decision trees that will compose the ensemble. For each inner node within a decision tree, the algorithm first randomly samples without replacement a subset of m attributes, and then it selects, among those attributes, the one that minimizes the local class impurity for that node. In this context, purity is the ratio of instances from difference classes that follow the same tree branch; hence, maximum purity in a node means that all instances reaching said node belong to the same class. This procedure is applied to each inner node in the current decision tree within the ensemble, and it is repeated until the tree achieves maximum class purity for all leaf nodes.

Random forests sometimes perform better than boosting methods, while being resilient to outliers and noise, faster to train than bagging and boosting methods (depending on the respective base learner), and being easily parallelized. However, it can require very many decision trees to provide an acceptable predictive performance, depending on the dataset at hand. Table 1 presents a brief overview of the main characteristics of the above traditional methods.

Table 1 Traditional ensemble learning methods compared. Adapted from [131].

Algorithm	Sampling	Base learner	Integration strategy
Bagging [18]	instance	Unstable learner trained over re-sampled instance subsets	Majority voting
Boosting [74]	instance	Weak learner re-weighted at every iteration	Weighted majority voting
Stacking [203]	None	Any	Meta-model
Random forests [19]	instance; attribute	Decision trees	Majority voting

3 Ensemble Learning with Evolutionary Algorithms

In general, Evolutionary Algorithms (EAs) are robust optimization methods that perform a global search in the space of candidate solutions. EAs are simple to implement, requiring little domain knowledge, and can produce several good solutions to the same problem due to their population-based nature [78]. In particular, EAs seem to be naturally suited for ensemble learning, given their capability of producing a set of solutions that can be readily integrated into an ensemble [120, 57]. EAs also support multi-objective optimization (based e.g. on Pareto dominance), allowing the generation of solutions that cover distinct aspects of the input space [120], and removing the need to manually optimize some hyper-parameters (e.g. the base learner's hyper-parameters, the number of ensemble members, etc.). However, EAs will likely increase the computational cost of ensemble learning, due to its robust global-search behavior that usually considers many tens or

hundreds of possible solutions at each iteration (generation). Nonetheless, parallelization is an option to mitigate such problem [123].

Since ensemble learning is composed of at least three main optimization steps (generation, selection, and integration), each one with many tasks, there is plenty of room to employ EAs [67]. In the literature on EA-based ensembles for supervised learning, the most common approach is to optimize a single step, though some studies go as far as optimizing two of them (e.g. [143, 29]). Figure 1 summarizes how many studies were dedicated to each of the three stages.

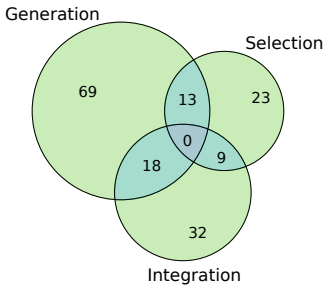

Figure 1: Work summarized by the ensemble learning stage that EAs are employed. While generation is more popular than selection and integration combined, none of the surveyed studies employed EAs in all stages.

In this work, we provide a taxonomy to categorize the EA-based approaches for supervised ensemble learning (Figure 2). All surveyed studies are focused on supervised problems, i.e., no unsupervised approach is reviewed.

We divide the surveyed studies according to the well-established main stages of supervised ensemble learning: generation, selection, and integration. We use these three stages at the top level of our proposed taxonomy because in principle major decisions about the design of the EA (e.g. which individual representation to use, which fitness function to use) are entirely dependent on the type of ensemble learning stage addressed by the EA. The approaches most often used in each stage are presented at the second level of the taxonomy. For example, attribute selection, model tuning, and instance selection are the three most common approaches for the generation stage. Further divisions in the taxonomy are presented at the next levels, whenever it is the case.

Note that taxonomies vary depending on the aspect being analyzed – e.g. Gu [81] is concerned with the generation stage, and hence proposes a taxonomy exclusively for that step. To the best of our knowledge, our taxonomy is one of the broadest with regard to EAs for ensemble learning, with the closest reference being the one proposed by Cruz et al. [39]. While the description of generation and selection stages in [39] is identical to ours, we are more specific regarding the strategies for the integration stage. In addition, while the authors propose a two-level taxonomy, we present a more detailed and thorough four-level taxonomy.

3.1 Methodology to collect and analyze papers in this survey

The main objective of this work is to identify and evaluate existing approaches that apply evolutionary algorithms for learning ensembles of predictive models for supervised machine learning. The objective is expressed from the research questions presented in Table 2. These questions aim to analyze the relevant work, both in the context of evolutionary algorithms used, and the characteristics of ensembles that are optimized. The sections where these questions are addressed are also shown in the table.

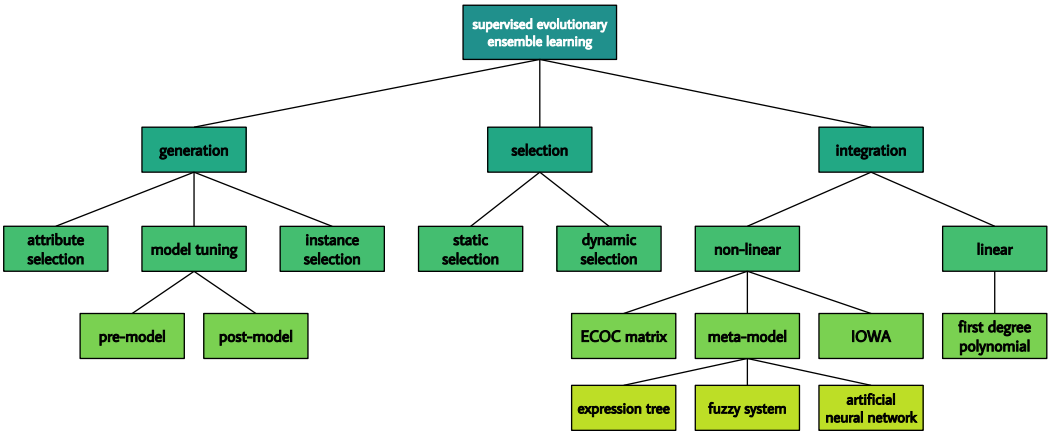

Figure 2: The proposed taxonomy for EAs employed in ensemble learning.

Table 2 Research questions of this survey.

ID	Research Question	Description	Addressed in
RQ1	What are the existing work that apply evolutionary algorithms for learning ensembles for supervised machine learning tasks?	General question that aims to identify existing work that apply evolutionary algorithms in the context of ensemble learning.	Throughout this survey
RQ2	What are the evolutionary algorithms used to learn the ensembles?	Aims to identify which evolutionary algorithms are applied for ensemble learning.	Section 8
RQ3	What stages of ensemble learning are addressed by the evolutionary algorithm?	Aims to categorize the approaches according to the ensemble optimization step (generation, selection or integration).	Sections 4, 5, and 6
RQ4	Which objective functions are optimized by the evolutionary algorithm?	Since fitness function is an essential component of EAs, and given the complexity of the ensembles where several objectives can be optimized, this question aims to analyze how these functions are employed in ensemble learning.	Section 7
RQ5	What are the base learners used?	Finally, this survey aims to analyze the relevant works from the point of view of the base learners that are used to compose the ensembles.	Section 9

3.1.1 Search Strategy

Based on the main objective, we select keywords that are likely to be present in most of the work that proposes evolutionary algorithms for ensemble learning; from these keywords we compose a search string. Synonyms of each term were incorporated using the Boolean operator *OR*, whereas the Boolean operator *AND* was used to link the terms. The generic search string derived is

```
‘ensemble’ AND
((‘classification’ OR ‘classifier’ OR ‘classifiers’) OR
(‘regression’ OR ‘regressor’ OR ‘regressors’)) AND
(‘evolutionary’ OR ‘evolution’)
```

In addition to the search string, we define the search engines. Thus, reviewed papers of this survey were searched in the following repositories: Scopus[1], Science Direct[2], IEEE Xplore[3], and ACM Digital Library[4]. Figure 3 shows the search strings as used in each search engine.

```
TITLE-ABS-KEY("ensemble") AND (
    (
        TITLE-ABS-KEY("classification") OR
            TITLE-ABS-KEY("classifier") OR
            TITLE-ABS-KEY("classifiers")
    ) OR (
        TITLE-ABS-KEY("regression") OR
        TITLE-ABS-KEY("regressor") OR
        TITLE-ABS-KEY("regressors")
    )
) AND (
    TITLE-ABS-KEY("evolutionary") OR
    TITLE-ABS-KEY("evolution")
)
```

(a) Scopus

```
"ensemble" AND (
    (
        "classification" OR
        "classifier" OR
        "classifiers"
    ) OR (
        "regression" OR
        "regressor" OR
        "regressors"
    )
) AND (
    "evolutionary" OR
    "evolution"
)
```

(b) ACM

```
"Abstract":ensemble AND (
    (
        "Abstract":classification OR
        "Abstract":classifier OR
        "Abstract":classifiers
    ) OR (
        "Abstract":regression OR
        "Abstract":regressor OR
        "Abstract":regressors
    )
) AND (
    "Abstract":evolutionary OR
    "Abstract":evolution
)
```

(c) IEEE Xplore

```
title-abstr-key("ensemble") AND (
    (
        title-abstr-key("classification") OR
        title-abstr-key("classifier") OR
        title-abstr-key("classifiers")
    ) OR (
        title-abstr-key("regression") OR
        title-abstr-key("regressor") OR
        title-abstr-key("regressors")
    )
) AND (
        title-abstr-key("evolutionary") OR
        title-abstr-key("evolution")
)
```

(d) ScienceDirect

Figure 3: Search strings as used in search engines.

3.1.2 Study Selection Criteria

We adopted the following criteria for including studies in this survey: (i) papers that present a new evolutionary strategy for ensemble learning in supervised machine learning; (ii) papers that present a minimum detail of the proposed solution, including: type of EA used, fitness function used, ensemble stage optimized, base learners used; and (iii) papers that present an experimental evaluation of the proposed solution. We also use exclusion criteria, which are: (i) unavailability: paper not available in any online repository, or papers available only under payment; (ii) wrong topic: on further review, papers that did not cover the surveyed topic; and (iii) papers that are not written in English.

3.1.3 Study Selection Procedures

In the selection process, the search string was applied to the title, abstract, and keywords of searched papers. Scopus was the first repository searched, since it has the largest database. Eight hundred and two (802) papers matched the keywords. All papers had their abstracts reviewed, and from their analysis the ones deemed relevant (366) were carried on to the next stage of the reviewing process, as shown in column **Relevant** of Table 3.

We proceed the search with ACM Digital Library, IEEE Xplore, and Science Direct, again reviewing abstracts and selecting papers based on their relevance to this survey. Since Scopus is the largest database, some papers present in the remaining repositories were also present in Scopus. For this reason, column **Already in Scopus** of Table 3 counts the number of papers

[1]Available at https://www.scopus.com/home.uri
[2]Available at http://www.sciencedirect.com
[3]Available at http://ieeexplore.ieee.org/Xplore/home.jsp
[4]Available at http://dl.acm.org

found in other repositories that already had their abstracts reviewed when we collected papers from Scopus. Among the remaining databases, we found 108 unique papers, not present in Scopus; from these, 38 were deemed relevant for further review.

Table 3 Papers that matched the search strings shown in Figure 3. In this stage all papers had their abstracts reviewed. From this initial analysis, not all papers were deemed relevant to the scope of this survey. **Already in Scopus** column denote papers that were already present at the Scopus database.

Repository	Relevant	Irrelevant	Already in Scopus	Total
Scopus	366	436	—	802
ACM Digital Library	16	11	34	61
IEEE Xplore	13	8	127	148
Science Direct	9	51	90	150
Total	**404**	506	251	1161

Across all searched databases, 404 papers were deemed relevant for the survey, taking into consideration the description of their abstracts. From these, 50 were unavailable, either because (i) the document was not found in their host websites, or it was behind a paywall; or (ii) on further review of the paper, the topic addressed was not exactly the one we were interested (43 papers), as discussed in the beginning of Section 3.1.2. This reduced the number of relevant papers to 311.

From the remaining 311 papers, 164 were fully reviewed and included in the survey, with the remaining 147 not included nor reviewed. While we could have reviewed the latter group, we did not because we applied a truncation factor: that is, the rate at which we were detecting new concepts in papers was not justified by the amount of papers needed to reach these novel ideas. The papers that were not reviewed did not have any characteristic that made them less attractive than the ones reviewed, and we do not discriminate based on vehicle of publication, type of publication (conference or journal paper), date, number of citations, etc. An overall summary of the papers is presented in Figure 4.

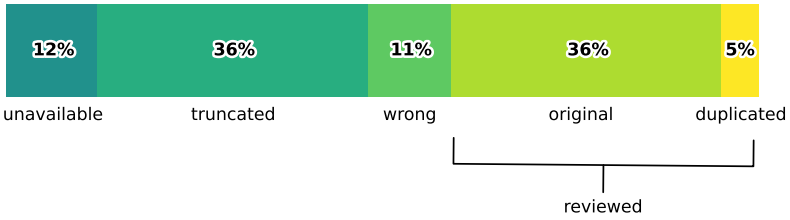

Figure 4: From the 404 papers selected for review, 164 were added to the survey. Among these, 20 were duplicated (e.g. expanded work), and 144 original work.

Papers were reviewed in chronological order: the ones closest to the date of the reviewing process were added first, and the ones that had been already been published, reviewed last. From the papers that made to the survey, 20 were duplicated and fell into one of the following categories: (i) the algorithms were published in conferences and had expanded versions in journals; (ii) they had different application domains, but the algorithm was the same; or (iii) slightly different implementations (for example, changing the number of layers and/or activations in a neural network).

3.1.4 Data Extraction Strategy

After selecting the 164 works to compose the survey, the extraction and analysis of the data was made through peer review, where at least two researchers evaluated each work. The data were structured in a spreadsheet according to its meta-data (catalog information) and the characteristics of the work, according to the research questions that we aimed to answer.

We make available two supplementary material to this paper. The first is a repository of source code, hosted at `https://github.com/henryzord/eael`, with metadata used to generate figures and tables in this paper. The other is a master table, made available as a website, and hosted at `https://henryzord.github.io/eael`, listing individual information on surveyed work.

4 The Generation Stage of Ensemble Learning

In this stage the algorithm generates a pool of trained models. Those models may come from: (i) different paradigms (e.g., Naïve Bayes, Support Vector Machines, and Neural Networks [157]); or (ii) be from the same paradigm, but still present some differences, such as artificial neural networks with different topologies and/or activation functions [213]).

The main objective in this stage is to generate a pool of both accurate and diverse base learners. Base learners must be diverse in order to provide source material for the selection and integration steps to work with. A diverse pool of base learners has more chances to commit errors in different data instances, thus correctly predicting more instances [150].

An example of an ensemble algorithm that focuses on the generation phase is random forests [19, 188], considering that it selects distinct subsets of both attributes and instances for building different decision trees, resulting in an ensemble of trees that is more robust than a single decision tree.

We have identified three ways of generating pools of learners that are both diverse and accurate: (i) providing distinct training sets for each base learner (instance selection); (ii) providing the same training set for all learners but with distinct sets of attributes (attribute selection); and (iii) optimizing the model by modifying the hyper-parameters and/or the structure of the base learners.

4.1 Instance Selection

Instance selection, also known as prototype selection or data randomization [196, 3] consists of providing different (not necessarily disjoint) subsets of training instances for different base learners [6, 167]. This approach is well-suited for homogeneous sets of base learners which are sensitive to changes in the instance distribution (e.g., decision trees [86]).

Instance selection can also be used to reduce training time by finding a subset of representative instances for each class [6, 167]. This is also beneficial for problems with high class imbalance, given that re-sampling instances with replacement makes it possible to simulate a uniform distribution among classes. Indeed that is one of the capabilities of the traditional bagging algorithm [18]. Thus, "bagged" EAs are likely to have the same benefits as "bagged": improved noise tolerance and reduced overfitting risk [196]. A method for selecting instances is needed since random sampling can lead to information loss and poor model generalization [95]. By using an EA, both the tasks of undersampling the majority class and oversampling the minority class are possible in parallel.

This section mainly focuses on instance selection techniques, since instance generation is more scarce. While the former performs a selection of instances from the original data, the latter can create new artificial instances, thus easing the adjustment of decision boundaries of models, at the expense of being more prone to overfitting [196]. Only one work uses a hybrid selection-and-generation strategy, namely [196]. In this work, seeking to address the class imbalance problem, a Steady State Memetic Algorithm (SSMA) is used for selecting instances from the training set, composing several individuals (i.e. subsets of instances). Once the SSMA optimization ends, a portion of its (fittest) individuals will be then fed to a Scale Factor Local Search in Differential Evolution (SFLSDE), that will improve the quality of instance subsets by generating new instances. Both evolutionary algorithms use a measure of predictive performance as the fitness function. Finally, the (fittest) individuals from SFLSDE are used by 1-NN classifiers, integrated by means of weighted voting (not evolutionary induced).

Although instance selection is present in work tackling the class imbalance problem (e.g. [196, 23, 77]), it can lead to overfitting, or having subsets where one class has many more instances than other classes, if an inadequate objective function (such as accuracy) is used. An approach to avoid overfitting is to assign different misclassification costs to different classes. Typical cost-sensitive learning techniques do not directly modify the data distribution, but rather take misclassification costs into account during model construction [23].

Instance selection can be divided into wrapper and filter methods. In our literature review, wrapper methods were more common than filter methods. Only the work of Almeida and Galvão [6] uses a filter approach, where a GA is used to optimize the number of groups in a k-means algorithm. Due to the tendency of k-means in finding hyperspherical clusters, the objective is to find evenly-distributed groups of instances. One classifier is trained for each group, and the quality of classifiers is assessed by the weighted combination of training accuracy, validation accuracy, and distribution of classes within groups. In the prediction phase, unknown instances will be assigned to their most similar cluster, based on the Euclidean distance between the unknown-class test instance and the cluster's instances.

For wrapper methods, usually the set of selected instances is encoded as either a binary or real-valued chromosome of N positions (the number of instances). In the binary case, each bit encodes whether an instance is present or absent in the solution encoded by the current individual. In a real-valued case, each gene encodes the probability that the respective instance will be present in that solution. To address class imbalance, in [77] only majority class instances are encoded in a binary string – minority class instances are always selected.

It is also possible to perform instance selection together with other methods. In [167], the multi-objective problem consists in optimizing both a Support Vector Machine's hyper-parameters and the instance set to be used for training each model. This combined strategy is better for SVMs than simply selecting training subsets, since SVMs are robust to small changes in data distribution [13]. Coupling two tasks at once also fits well with weight optimization: in [112], evolutionary under-sampling and boosting are used in a C4.5 decision-tree classifier to iteratively optimize its performance in grading breast cancer malignancy.

Olvera-López provides an extensive survey of both evolutionary and non-evolutionary instance-selection methods proposed until 2010 [145].

4.2 Attribute selection

Attribute selection, also called feature selection or variable subset selection [176], offers distinct subsets of attributes to different base learners in order to induce diversity among base models. By removing irrelevant and redundant attributes from the data, attribute selection can improve the performance of base learners [179]. Reducing the number of attributes also reduces the complexity of learned base models, and may improve the efficiency of the ensemble system.

Attribute selection also performs dimensionality reduction, and is an efficient approach to build ensembles of base learners [126]. There is no need to provide disjoint sets of attributes to different learners. The base learners must be sensitive to modifications in the data distribution. Support Vector Machines, for example, were reported to be little affected by attribute selection [193].

Wrapper methods are by far the most common type of EAs for attribute selection. It has been noted that there is a direct link between high-quality attribute subsets and a high-quality pool of base learners [135]. Filter approaches break this link, evaluating the quality of an attribute set in a way independent from the overall base learner pool [135].

A wrapper method provides a reduced subset of attributes to a learning algorithm, and then the predictive accuracy of the model trained with those attributes is used as a measure of the quality of the selected attributes. The random subspace method, for example, is a traditional approach for wrapping algorithms that randomly selects different attribute subsets for different base learners. Although this method is usually much faster than EAs, its performance is sensitive to the number of attributes and ensemble size [126]. By contrast, EAs can improve stability and

provide more accurate ensembles [126]. Other examples of traditional methods include sequential forward selection, sequential backward selection, beam search, etc. [135].

However, filter methods are still usually preferred for some application domains, such as microarray data, where the number of attributes far surpasses the number of instances, rendering a wrapper approach inefficient. In [98], base learners are coupled with filters that perform attribute selection. Although the authors do not use training time as an objective in the EA, the reported execution time of a single run of the GA is between 15 seconds to 3 minutes, much faster than an exhaustive search, that could take as long as one hour (for sets of 24 attributes), or one year (for sets of 42 attributes), in a dataset of 4026 attributes. Their proposed algorithm is also capable of outperforming other EA-based ensembles for two microarray datasets. In another study, genetic algorithms (GAs) with error rate as fitness function were shown to be capable of outperforming greedy wrapper methods in terms of ensemble accuracy [135].

Two concepts relevant for attribute selection are sparsity and algorithmic stability. An attribute selection algorithm is called sparse if it finds the sparsest or nearly-sparsest set of attributes subject to performance constraints (e.g. small generalization error) [193]. An algorithm is called stable if it produces similar outputs when fed with similar inputs – i.e., it selects similar attribute sets for two similar datasets [208]. As noted in [193, 208], stability and sparsity constitute a trade-off. An algorithm that is sparse may be incapable of selecting similar sets of attributes across runs [193].

EAs for attribute selection vary on the number of objectives to be optimized, integration with other stages, and distribution of base learners. In [156, 157] a multi-objective Particle Swarm Optimization algorithm provided different attribute subsets to heterogeneous base learners, in order to predict whether power transforms will fail in the near future. In [181, 179, 180], two Pareto-based multi-objective differential evolution algorithms performs attribute selection, and then linear voting weight optimization, in a pipeline fashion (the generation stage is performed before the integration stage).

The encoding used in [101] considers each individual as an ensemble of classifiers. Classifiers are trained differently based on their input features. Each classifier competes with its neighbors within the same ensemble; and at a higher level, ensembles compete among themselves based on their predictive accuracy.

4.3 Model optimization

Models may have their hyper-parameters and/or structure modified while creating a pool or ensemble of base learners. We divide this category of our taxonomy into two groups: pre-model and post-model optimization.

Pre-model optimization involves fine-tuning the hyper-parameters of the base learners that will generate the base models. We call these approaches *pre-model* because the optimization happens prior to model generation. Examples are: tuning a neural network's learning rate; a SVM's type of kernel function [167]; L2 regularization [205]; and random forests' number of trees [169].

Pre-model approaches may support heterogeneous sets of base learners. E.g., in [169] the authors select both the types of base learner and their respective hyper-parameters, together with a set of attributes that will be assigned to a given learner. They used the NSGA-II [49] algorithm, and the one-point crossover keeps base models and hyper-parameters together, only allowing to swap the selected attributes for each model.

Post-model approaches try to improve an existing model. Examples are layout and inner node selection for decision trees [134, 9, 199], and topology, weight, and activation function optimization in neural networks [67, 143, 143]. Weights are also optimized in [114], where an ensemble of heterogeneous parametric models are optimized by differential evolution.

Post-model encoding depends on the type of base learner being used, and so are more common on homogeneous sets of base learners. In [99], the weights of artificial neural networks are modified by a GA. The authors adopt a matrix of size $W \times W$, where W is the number of neurons in the

entire network. The upper diagonal encodes whether two given neurons are connected, and the lower diagonal encodes the weights associated with those connections.

Some studies perform both pre- and post-model optimization. In [143], first the topology of a neural network is evolved by using NSGA-II. The best found topology then has its parameters (e.g., weights and activation functions) adjusted by a multi-objective Differential Evolution method. In the end, the final population is submitted to a voting scheme optimized by another EA.

Attribute selection is often coupled with model optimization. In [187], both post-model optimization of Radial Basis Function Neural Networks and attribute selection were used, by performing both approaches in two subpopulations of the Cooperative Coevolutionary EA. In [162], solutions for both tasks were placed within the same chromosome: in a 132-wide chromosome array, 88 bits are designated for attribute selection, 10 bits represent parameter *nu* and threshold (integer and decimal part), 14 bits correspond to the gamma value and 20 bits are used for the parameter C of a nu-SVR learner.

Table 4 summarizes the work on EAs for the generation step of supervised ensemble learning, based on the taxonomy proposed in this section.

Table 4 Categorization of studies that employ EAs in the generation stage of supervised ensemble learning.

Method	Related work
Instance selection	[112, 108, 6, 81, 95, 43, 77, 196, 2]
Attribute selection	[111, 157, 156, 181, 169, 140, 29, 81, 123, 180, 200, 179, 176, 178, 177, 101, 125, 135, 33, 211, 193, 187, 162, 52, 50, 116, 5, 10, 185, 28, 44, 42, 13, 40]
Pre-model optimization	[205, 167, 169, 6, 143, 96, 123, 200, 35, 36, 150, 37, 93, 94, 166, 193, 162, 53, 182, 44, 42, 129, 122, 13]
Post-model optimization	[114, 67, 199, 134, 9, 143, 30, 120, 41, 119, 121, 99, 71, 69, 70, 68, 57, 47, 183, 25, 61, 63, 62, 64, 132, 133, 164, 79, 190, 130, 43, 87, 24, 23, 171, 187, 102, 174, 16, 17, 194, 52, 50, 92, 163, 66, 182, 58, 44, 42, 197]

5 The Selection Stage of Ensemble Learning

From the pool of generated base models, model selection (or model pruning [151]) is performed in order to define the final set of base models for the ensemble. Selection may be regarded as a multi-objective problem, where two objectives – predictive performance, and diversity – must be optimized. When the size of an ensemble is large, selecting base models based on these metrics can be computationally expensive if all ensemble options are considered, thus making the use of meta-heuristics (such as evolutionary algorithms) attractive [151]. However, simpler options (such as simply selecting the Φ most accurate learners) can also be used. Selection is often viewed as an optional stage and frequently not performed by traditional methods (e.g., boosting [74], bagging [18]) or EA-based ones (e.g., [24, 211]).

Whether or not to perform selection is an issue for debate, with some authors proposing to bypass this stage (i.e., using the entire pool of models as ensemble) [189]. Lacy et al. [120] argue that model selection is irrelevant for ensemble learning, and that it is sufficient to select the Φ best models from the pool. According to [120, 75, 21], from a predictive performance standpoint, this approach would be more effective than building an ensemble while considering diversity metrics. Other authors claim that there is little correlation between ensemble diversity and accuracy [148, 19, 151, 21]. On the other hand, some authors argue the opposite: e.g., for regression, Wang and Alhamdoosh [198] argue that the Φ best neural networks may not produce an ensemble with better Mean Squared Error (MSE). This is also stated by Liu et al. [127], adding that simply selecting the most accurate models may result in loss of predictive performance given that most of those models may be strongly correlated, leaving the opinion of the minority of the committee underrepresented.

Although Lacy et al. [120] and Liu et al. [127] have different opinions on the utility of model selection, both agree that diversity measures are not a good proxy for ensemble quality, with Liu et al. [127] suggesting that accuracy on a validation[5] set is sufficient. The rationale for using diversity measures is that by sacrificing individual accuracy for group diversity, one can achieve better group accuracy [151, 25]. Diversity in this case should not be measured at the genotype level (e.g., individuals encoding different attributes for the same base model), but rather measured based on the predictive performance of the algorithms decoded from the individuals. Diversity metrics can be of two types: pairwise or group-wise [86]. A pairwise diversity metric often outputs a matrix of values denoting how diverse one base model is from another. Then, algorithms may select models that are, e.g., more diverse to the other already-selected models. By contrast, group-wise metrics validate how diverse a group of base models is, thus requiring a previous strategy for composing groups. A review of diversity measures is presented by Hernández et al. [86].

The motivations for using EAs for model selection are as follows. First, finding the optimal model subset within a large set is unfeasible with exhaustive search (the search space size is $\approx 2^B$, where B is the number of base models). By contrast, EAs perform a robust, global-search for the near-optimal set of base models [151]. There is evidence that smaller ensembles can indeed outperform larger ones [188]. However, in practice, the optimal ensemble size varies across types of ensembles (e.g. bagging vs boosting), types of base learners (with different biases), and datasets (with different degrees of complexity). Hence, given the complexity of the problem of selecting the optimal model size, and the typically large size of the search space, it is justifiable to use a robust search method like EAs to try to solve this problem.

Model selection can be further divided into two categories: static and dynamic selection [43, 90, 89, 39]. In static selection, regions of competence are defined at training time and are never changed [90, 89, 39]. In dynamic selection the regions are defined during classification time, through the use of a competence estimator [90, 89, 192]. Figure 5 puts both strategies in perspective.

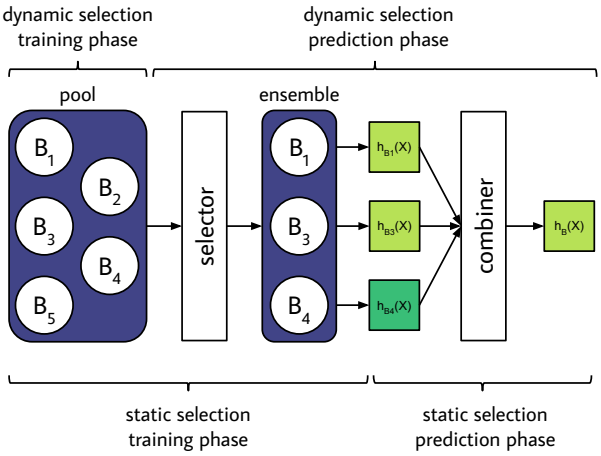

Figure 5: Difference between static and dynamic selection strategies. While in static selection the competence estimator assigns regions to base learners during the training phase, in dynamic selection this is done during the prediction phase. Dynamic selection can also have a selector (e.g. oracle) that assigns a single base learner to regions of competence.

Some studies in the literature (e.g. [20, 39]) experimentally assess whether dynamic selection methods are better than static selection ones. In [39], the authors compare static and dynamic

[5]In supervised learning it is common to divide a dataset into three disjoint sets: training, validation and test. The validation set is used to evaluate the quality of models *while* training, and helps to prevent overfitting to the training data. The test set is used for the final model evaluation *after* training.

selection methods on 30 different datasets, under the same protocol. The authors also compare these strategies with well-established ensemble algorithms, such as random forests, and AdaBoost. Only one of the 18 dynamic selection methods presented a worse predictive accuracy than simply using the best-performing classifier in the ensemble, and 66% of them outperformed a genetic algorithm performing static selection with majority voting as integration strategy. Furthermore, 44% and 61% of them presented a better average ranking than Random Forests and AdaBoost, respectively.

These results are also supported by [20]. Dynamic selection methods were statistically better than three other strategies: using the single best classifier in an ensemble, using all the generated classifiers, and using static selection methods. For the latter, dynamic selection algorithms won in 68% of the cases.

Note that some studies say they perform dynamic selection (e.g. [6] via k-means), but in fact they perform static selection, since the assignment of classifiers is done during training time and does not change after that.

5.1 Static selection

Static selection is well-suited for batch-based learning, where the data distribution is not expected to change with time. Static selection assigns regions of competence during training time, thus allowing some freedom regarding which methods can be used. *Overproduce-and-select* is a traditional strategy for ensemble learning [93, 38], where an algorithm first generates a large pool of base models using a generation method (see Section 4). Then, the base models are selected from this pool and their votes are combined via an integration scheme (see Section 6). The rationale is that some models may perform poorly or have strongly-correlated predictions, making some of these safe for exclusion from the final ensemble [189]. EAs for this strategy aim at selecting the set of models that optimize a given criterion(a), often used as the fitness function.

A second strategy for static selection, known as *clustering-and-selection*, uses a clustering algorithm to assign models to distinct regions of competence in the training phase. In the testing phase, a new instance is submitted to the base model that covers the region closer to that instance. Studies using this strategy include [160, 159, 59]. In [90], a GA was used for selecting the number of partitions in which the input space is divided. It then assigns an ensemble of classifiers to each partition, optimizing the voting weight of each base learner.

Overproduce-and-select and *clustering-and-select* differ regarding the region of competence where they will be employed. In the former, all selected base models will cast their predictions over the same region, whereas in the second they will be assigned to distinct ones.

In [198] a hill-climbing strategy was used for increasing the size of the ensemble. By starting with only two classifiers (Neural Networks), the number of ensemble members is increased by adding classifiers that reduce the overall ensemble's error rate. In [56], the authors investigate the impact of combining error rate (effectiveness), ensemble size (efficiency), and 12 diversity measures on the quality of static selection by using pairs of objectives. The authors also study conflicts between objectives, such as error rate/diversity measures and ensemble size/diversity measures. They argue that, among diversity measures, difficulty, inter-rate agreement, correlation coefficient, and double-fault are the best for combining with error rate, ultimately producing the best ensembles.

Studies that use the overproduce-and-select strategy often encode individuals as binary strings, where 0 denotes the absence of that model in the final ensemble and 1 the presence [29]. However, in [158] the individual size was doubled by using two values for each model: one for the aforementioned task, and another to determine the strength of that model's output in the final ensemble's prediction.

In [98], attribute and model selection were performed at the same time. The authors use a binary matrix chromosome where each row represents a different base learner and each column a filter-based attribute selection approach. In this sense, if a bit is active somewhere within the

individual's genotype, it means that the base learner of the corresponding row will be trained with the attributes selected by the filter approach of the corresponding column.

5.2 Dynamic selection

In dynamic selection, a single model or a subset of most competent learners is assigned to predict an unknown-class instance [39] (hereafter, unknown instance for short). This strategy is better suited for e.g. data stream learning, since the competence estimator naturally assigns base models to instances during the prediction phase. Dynamic selection was reported to perform better than boosting and static selection strategies [39]. However, work on dynamic selection is much less frequent than work on static selection. Dynamic selection is also more computationally expensive, since estimators are required to define regions of competence for all predictions, which can be unfeasible in some cases [44, 20].

One approach for dynamic selection is to use random oracles [38, 188]. A random oracle is a mini-ensemble with only two base learners that are randomly assigned to competence regions [188]. At prediction time, the oracle decides which base learner to use for providing predictions for unknown instances.

Another strategy is to train a meta-learner. In [123], generation strategies of feature selection and pre-model optimization were combined with an overproduce-and-select strategy for generating a diverse pool of base learners. Next, a meta-learner was trained for selecting the best subset of models for predicting the class of unseen instances.

Table 5 shows the distribution of the surveyed EAs into the static and dynamic selection categories. The interested reader is referred to the work of Cruz et al. [39] for a review on dynamic selection strategies.

Table 5 Categorization of EAs for the selection stage of supervised ensemble learning.

Method	Related work
Static selection	[6, 137, 181, 169, 11, 158, 139, 140, 29, 151, 152, 100, 90, 89, 104, 55, 56, 98, 97, 154, 153, 34, 166, 86, 27, 25, 131, 38, 116, 198, 186, 182, 32, 58, 46, 160, 159, 59, 175, 122, 7, 13, 12]
Dynamic selection	[123, 38, 188]

6 The Integration Stage of Ensemble Learning

The last step of ensemble learning concerns the integration of votes (for classification) or value approximation (for regression) in order to maximize predictive performance. Ensemble integration, also called learner fusion [189] or post-gate stage [51], is the final chance to fine-tune the ensemble members in order to correct minor faults, such as giving more importance to a minority of learners that are however making correct predictions. Integration is an active research area in ensemble learning [189]. Similarly to the selection stage, this is another stage where using a validation set can be useful, since reusing the training set that was employed to generate base models can lead to overfitting.

As with other ensemble stages, there are two approaches for the integration of base learners: using traditional, non-EA methods, or using evolutionary algorithms. For classification, the most popular method is weighted majority voting, which allows to weight the contribution of each individual classifier to the prediction according to its competence via voting weights [189]:

$$h_B(X^{(i)}) = \underset{j}{\operatorname{argmax}} \left(\sum_{b=1}^{B} w_{b,j} \times [h_b(X) = c_j] \right) \tag{1}$$

where B is the number of classifiers, $w_{b,j}$ the weight associated with the b^{th} classifier for the j^{th} class, and $[h_b(X) = c_j]$ outputs 1 or 0 depending on the result of the Boolean test. This strategy has been shown to perform better than majority voting and averaging [120]. A simplification

of that function sets all weights to 1, which turns this method into a simple majority voting, another popular approach [214]. For instance, bagging uses a simple majority voting scheme, whereas boosting uses weighted majority voting [215].

For regression, the most popular is the simple mean rule, which averages the predictions of base regressors, $h_B(X^{(i)}) = \frac{1}{B}\sum_{b=1}^{B} h_b(X^{(i)})$, where B is the number of regressors and $h_b(X^{(i)})$ is the prediction for the b^{th} regressor. Simple aggregation strategies are better suited for problems where all predictions have comparable performance, however those methods are very vulnerable to outliers and unevenly-performing models [131]. Other traditional methods for integrating regressors are average, weighted average, maximum, minimum, sum, and product rules [103, 135, 119].

However, there are plenty of studies that employ EAs for integrating base learners. These studies can be divided into two categories: optimizing the voting weights of a weighted majority voting rule; or optimizing/selecting the meta-models that will combine the outputs of base learners. Both categories may be interpreted as using meta-models for this task, as in stacking [192, 135]. Ensembles that use stacking are referred to as two-tier (or two-level) ensembles [192]. Those ensembles are well-suited, e.g., for incremental learning [192]. Actually, when updating an existing ensemble model to consider new data, we may need to train only a few novel base models covering the new data and then re-train the meta-model with the both the novel and the previous base models. This is more efficient than re-training all existing base models in a single-level ensemble [192]. Two-tier ensembles were reported to perform better than simple weighting strategies in larger datasets [138]. As disadvantages, two-tier ensembles are more susceptible to overfitting when compared to traditional integration methods, and also increase the training time of the entire ensemble [131]. In practice, whether stacking or traditional aggregation methods are better is heavily influenced by the input data [138]. The following sections will review the available methods that use EAs for the integration of base learners.

6.1 Linear models

Evolutionary algorithms in this category are concerned in learning a set of voting weights that will be used in a weighted majority voting integration strategy. A wide variety of methods were proposed for this task, such as using genetic algorithms [143, 112], particle swarm optimization [170], flower pollination [213], differential evolution [215, 216, 176], etc. Those methods can be applied to both homogeneous [215, 216, 26] and heterogeneous [100, 142, 214] base learner sets. For classification, methods may also differ in the number of voting weights, either by using one voting weight *per* classifier (e.g. [214, 139]) or one voting weight *per* classifier *per* class (e.g. [65, 180, 41]).

For a thorough experimental analysis of both linear and non-linear voting schemes, the reader is referred to the work of Lacy et al. [119], which presents the most comprehensive experimental comparison of EA-based combining methods to date. Notwithstanding, in the next sections we present a broader review of EAs proposed for this task, as well as methods that were not presented in [119].

6.2 Non-linear models

Instead of optimizing weights, one can use non-linear models for integrating predictions. As the name implies, a non-linear integration model does not use a set of voting weights (one for each model) to cast predictions, but instead relies on another arrangement to combine votes. As it is shown in this section, the types of non-linear integration models used in literature may range from neural networks, to expression trees. Nonetheless, these non-linear integration models may better exploit classifiers' diversity and accuracy properties [60].

6.2.1 Expression trees

One of the most popular EA-based non-linear methods are expression trees [72, 120, 119, 4, 127, 124, 191, 60]. Expression trees have models in their leaves and combination operators in their inner nodes.

For the problem of microarray data classification, in [127, 124] some decision trees (initially trained with bagging) are fed to a Genetic Programming algorithm, which then induces a population of expression trees (each allowed to have at most 3 levels) for combining the base classifiers' votes. After the evolutionary process is completed, expression trees with accuracy higher than the average are selected by a forward-search algorithm to compose the final meta-committee, which will predict the class of unknown instances.

6.2.2 Genetic Fuzzy Systems

Genetic fuzzy systems are popular in ensemble learning, where fuzzy systems optimized by EAs are used to predict the class of unknown instances. A study reports that fuzzy combiners can outperform crisp combiners in several scenarios [189]. There are several steps in the induction of fuzzy systems where EAs may be used: from tuning fuzzy membership functions to inducing rule bases [38, 192]. For instance, in [38, 189], a GA was used with a sparse matrix for codifying features and linguistic terms; and in [192] a GP algorithm was used to evolve combination structures of a fuzzy system.

6.2.3 Neural Networks

In an empirical work comparing several integration methods [119], a multilayer percetron was used as a combination strategy. The output from base classifiers was used as input for the neural network, with an EA used for optimizing the weights of connections between neurons.

6.2.4 Evolutionary Algorithms for selecting meta-combiners

In [175], besides using the Artificial Bee Colony (ABC) algorithm for selecting base classifiers, the authors also use another ABC for selecting the meta-learner that will combine the votes of ensemble members.

6.3 Other methods

6.3.1 Induced Ordered Weighted Averaging (IOWA)

Ordered Weighted Averaging (OWA) [209] is a family of operators designed to combine several criteria in a multi-criteria problem. Let $A_1, A_2, A_3, \ldots, A_z$ be z criteria to be fulfilled in a multi-criteria decision function, and let A_j be how much a given solution fulfills the j-th criterion, $A_j \in [0, 1], \forall j = 1, \ldots, z$. The problem is then how to measure and compare solutions. This is solved by employing the OWA operators. OWA will combine two sets of values, a set of weights $W_1, W_2, \ldots, W_z, W_j \in (0, 1), \forall j = 1, \ldots, z, \sum_{j=1}^{z} W_j = 1$, and the set of ordered criteria $B = decreasing\_sort(A)$, by using a dot product, $F(A) = \sum_{j=1}^{z} W_j B_j$, with $F(A)$ as the fulfillment score of the solution. OWA is deemed ordered because weights are associated with the **position** in the combination function, rather than a specific criterion. For performing the combination, the criteria A are ordered based on their fulfillment rate (that is, the criterion that was most satisfied is combined with the first weight; the second most fulfilled criterion is combined with the second weight; and so on).

A method called Induced Ordered Weighted Averaging, or IOWA, is concerned with inducing the set of weights W, based on observational data (e.g. a dataset). In [15] a Multi-Objective EA based on Decomposition (MOEA-D) is used for inducing these weights, and IOWA is used to combine predictions from a set of Gaussian Process Regressors (GPR).

6.3.2 Error Correcting Output Codes (ECOC)

Error Correcting Output Codes (ECOC) [14] is a meta-method which combines many binary classifiers in order to solve multi-class problems [10]. It is an alternative to other multi-class strategies for binary classifiers [22] – such as one-vs-one, which learns a classifier for each pair of classes; and one-vs-all, which learns one classifier *per* class, discriminating instances from that class (positives) from all other instances (negatives). ECOC provides meta-classes to its classifiers (i.e. positive and negative classes are in fact combinations of instances from one or more classes). An example of ECOC is shown in Figure 6.

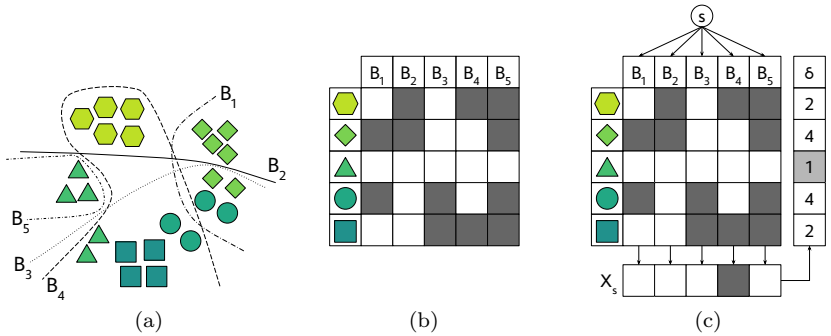

Figure 6: (a) Feature space and decision boundaries of base classifiers. (b) Coding matrix, where black and white cells correspond to positive and negative classes, respectively, denoting the two partitions to be learned by each base classifier. (c) Decoding step, where the classifiers' predictions $\{b_1, b_2, \ldots, b_5\}$ for a given sample s are compared to the codewords $\{y_1, \ldots, y_N\}$ and s is labeled as the class codeword at minimum distance. Adapted from [14].

ECOC comprises two steps: encoding and decoding. The aim of encoding is to design a discrete decomposition matrix (codematrix) for the given problem [10]. A study reports that larger matrices (with regard to number of classifiers) improve predictive performance [10]. In the decoding phase, each classifier casts a vote to a meta-class for an unknown instance. The predicted class is computed by comparing the distance of the outputed codeword for that instance with the codeword from each real class via a similarity metric.

Though in classification we wish to reach top predictive accuracy, other measures should also be considered for evaluating the ECOC matrix, such as row separation and column diversity [22]. By using an indicator-based selection EA (IBEA), in [22] the ensemble accuracy, individual classifier accuracy, and hamming distance were used as objectives for optimizing the layout of ECOC matrices, by manipulating the distribution of classes among base classifiers. In [10], on the other hand, an attribute selection strategy was used to generate classifiers to be integrated by an ECOC scheme; hence, this work is labeled as a generation technique instead of an integration one.

To summarize, Table 6 shows the categorization of studies using EAs in the integration stage of ensemble learning, based on the type of integration method.

Sections 4, 5, and the current section 6 have discussed EAs for each of the three stages of ensemble learning. In the next two sections, we focus on broader aspects of EAs for ensemble learning that are not specific to any single stage.

7 Objective (or fitness) functions

The fitness function is an essential component of an EA, since it defines the objective(s) to be optimized and guides the search accordingly. Due to the complexity of ensemble learning, there are several types of objectives that can be optimized. In this section we first review separately

Table 6 Categorization of studies using EAs in the integration stage of ensemble learning.

Method	Related work
First Degree Polynomial	[212, 83, 215, 216, 112, 111, 108, 109, 107, 110, 26, 139, 213, 146, 158, 170, 11, 143, 41, 119, 100, 180, 179, 90, 88, 89, 141, 142, 172, 206, 176, 178, 177, 98, 125, 61, 64, 63, 62, 76, 214, 128, 65, 91, 77, 21]
Expression trees	[72, 120, 119, 4, 127, 124, 191, 60]
Genetic Fuzzy System	[38, 189, 192]
Error Correcting Output Codes	[22]
Artificial Neural Network	[119]
IOWA	[15]
Meta-learner selection	[175]

each of four broad types of objective (fitness) functions: effectiveness, efficiency, diversity, and complexity. Next, we review multi-objective optimization approaches.

7.1 Effectiveness, Diversity, Complexity and Efficiency

An objective function measures *effectiveness* when it evaluates the ensemble's predictive accuracy. This is essential to ensemble learning and is addressed by all surveyed studies. The most popular objectives within this category are accuracy (or its dual, error rate) for classification tasks and mean squared error for regression tasks. The well-known accuracy measure is given by:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

where TP, TN, FP and FN are the numbers of true positives, true negatives, false positives, and false negatives, respectively. The error rate is simply: $1 - accuracy$. Note that accuracy and error rate have the drawback of not being suitable for highly imbalanced class distributions [121], since they are relatively easy to optimize by predicting nearly always the majority class.

The mean squared error is given by:

$$\text{MSE}(X^{(i)}) = \frac{1}{N}\sum_{i=1}^{N}(h_B(X^{(i)}) - Y^{(i)})^2 \tag{3}$$

which computes the difference between the predicted value $h_B(X^{(i)})$ and the real value $Y^{(i)}$, for all N instances. Other effectiveness measures include exponential squared loss [91], geometric mean [196, 24, 162, 23], imbalance ratio [196], and confidence [182, 100, 98], to name just a few. In general, such measures have the advantage of coping better with imbalanced class distributions than the aforementioned accuracy measure (or its dual error rate).

A diversity metric evaluates how diverse an ensemble's members (base learners) are. A diversity measure is often used as an objective in the selection stage of ensemble learning (Section 5); and it can also used in the generation stage (Section 4). We refer the reader to [25] for a review on diversity measures for generating models; and next we discuss the controversial issue of using diversity as an objective in general, regardless of the ensemble learning stage.

Several researchers defend diversity as a valid objective (e.g. [68, 31, 183, 29]), stating that it contributes to ensemble accuracy [31]. De Stefano et al. [183] state that, as the number of base learners increase, so does the probability that a minority of correct base learners will be overrun by a majority of wrong base learners, and thus the need for using diversity measures to reverse that effect. Also, in EAs, genetic material from well-performing solutions tend to be propagated to their offspring, often compromising diversity [57].

However, other researchers do not see the utility of diversity measures (e.g. [120, 127, 104]), stating that the correlation between ensemble accuracy and diversity is not as strong as expected [188]. Some authors also note that classic ensemble learning methods (e.g. bagging, boosting, and random subspace) introduce diversity in an ensemble without directly measuring

it [56]. We can conclude, from this debate, that the relationship between ensemble effectiveness and diversity is not fully understood yet [56, 188].

Diversity can be measured based on the ensemble's characteristics encoded in an individual's genotype, or based on the predictions made by each base learner. In the latter case, a set of base learners is said to be diverse when their errors are not correlated [170]. Examples of diversity measures in this category include Yule's Q statistic [187], average residual correlation coefficient [191], and negative correlation [17, 16]. The most popular measure seems to be the Kohavi-Wolpert variance [160, 159, 32], given by:

$$KW = \frac{1}{NB^2} \sum_{j=1}^{N} L(X^{(j)})(B - L(X^{(j)})) \tag{4}$$

where B is the number of classifiers, N is the number of instances in the (training or validation) evaluation set, and $L(X^{(j)})$ is the number of classifiers within the ensemble that correctly predict the class of instance $X^{(j)}$.

Diversity measures can be divided into pairwise and group measures. The latter evaluate diversity among all classifiers in the ensemble, whereas pairwise metrics evaluate diversity between two classifiers, and require an averaging technique for obtaining a group measure from all classifier-pairwise measures [86]. This is performed by the disagreement measure. The pairwise disagreement measure [25] is defined as:

$$\text{Diff}(B_i, B_j) = \frac{L^{01} + L^{10}}{L^{00} + L^{01}L^{10} + L^{11}} \tag{5}$$

where B_i and B_j are respectively the i-th and j-th classifiers within the ensemble, L^{10} is the number of instances correctly classified by B_i and wrongly classified by B_j, and so on for the remaining indices L^{01}, L^{00}, L^{11}. Pairwise disagreement varies from 0 to 1, with 0 indicating no disagreement (i.e. equal predictions) and 1 maximum disagreement. The plain disagreement measure [125] simply averages the overall disagreement among the members of the ensemble:

$$\text{PSM} = \sum_{i=1}^{B} \sum_{j=i+1}^{B} \sum_{k=1}^{N} \frac{\text{Diff}(B_i, B_j)}{((B-1) \times B \times N)} \tag{6}$$

where B is the number of classifiers, N the number of instances in the training or validation set. For an extensive list of diversity measures for ensemble learning, the reader is referred to [118, 86, 104].

Complexity metrics evaluate how complex the classifiers in the ensemble, or the ensemble as a whole, are. The most popular complexity metrics are the number of activated classifiers (for classifier selection) [189, 188, 38, 98, 56, 154, 87, 97] and the number of attributes used by the models induced by the base learners [123, 180, 200, 185, 28, 211, 162, 5, 27]. Other complexity metrics include the number of nodes in flexible neural trees [143]; the number of hidden neurons in a neural network [123, 37]; the structured minimization principle [79]; the number of support vectors in a Support Vector Machine model [162]; and the length of fuzzy rules [87]. Most of these are measures of the size of an ensemble or its base members, so they are simple to compute; but the trade-off is that they may not capture a more sophisticated aspect of complexity (like complex interactions between the ensemble's base members).

Efficiency is a desired objective when an ensemble must be fast, during training and/or testing (prediction) phase. Training efficiency is obviously important in very large datasets. In addition, both training and testing efficiency are especially important in data stream scenarios, where a continuous flow of incoming data is presented to the system and predictions must be made in a real-time basis. As a fitness function, training and test time also have the advantage of being very easy to compute; but they can introduce a trade-off between computational time and effectiveness, which could reduce the EA's effectiveness.

Complexity and efficiency metrics are related since, broadly speaking, reducing the complexity of the base learners or the ensemble as a whole leads to more efficient ensemble learning systems – e.g., reducing the number of base learners (a complexity metric) leads to faster ensembles, for a fixed type of base learner. Note, however, that the number of base learners is not directly a measure of efficiency, since efficiency depends on both the number and the type of base learners. For example, an ensemble with a given number N of decision tree algorithms would probably be trained faster than an ensemble with $N/2$ neural networks, since the latter type of base learner is much slower than the former. In addition, it is possible to improve efficiency without directly reducing the complexity of the models in the ensemble – e.g., by reducing the number of instances fed to the ensemble learning system.

In our survey, only two studies optimize efficiency, one measuring prediction time reduction [140], and the other measuring training set size reduction (for instance selection) [167], as a proxy for training time. None of the surveyed EAs employed training time *per se* as a metric.

7.2 Single vs. Multi-Objective Optimization

Ensemble learning methods performing single-objective optimization are obviously constrained to optimize effectiveness. However, ensemble learning may be naturally viewed as a multi-objective task, involving also other types of objectives. Figure 7 shows the distribution of other objectives that were optimized along effectiveness in studies that employed multiple objectives.

In this work we follow the taxonomy of multi-objective optimization approaches proposed in [73], where approaches are categorized into three types: (i) weighted fitness functions, where each objective is assigned a user-defined (typically, very ad-hoc) weight indicating that objective's importance; (ii) the lexicographic approach, where the user only ranks the objectives in terms of their priorities (no ad-hoc numerical weights), and then the EA selects individuals for reproduction by trying to optimize the objectives in their decreasing order of priority; and (iii) the Pareto dominance approach, where the EA evolves a set of non-dominated solutions in the Pareto sense – i.e., a solution is non-dominated if it is not worse than any other according to each objective *and* it is better than others according to at least one objective.

In the surveyed papers, the least popular approach was the lexicographic one ([123]), followed by weighted fitness functions ([212, 151, 86, 152, 100, 200, 98, 97, 154, 34, 122, 13]), then the single-objective approach (see the single-objective entry in Table 7) and finally the Pareto dominance approach as the most popular one (all the papers that were not cited in this paragraph and are within the multi-objective entry in Table 7).

Table 7 Studies categorized by number and type of objectives employed.

Number of objectives	Nature	Related work
Single-objective	Effectiveness	[26, 114, 213, 83, 6, 72, 96, 29, 41, 119, 100, 95, 4, 127, 124, 109, 107, 110, 90, 89, 141, 142, 172, 206, 55, 35, 36, 150, 93, 94, 176, 177, 101, 153, 99, 71, 69, 70, 68, 125, 57, 47, 183, 61, 63, 62, 64, 132, 133, 164, 131, 33, 190, 76, 214, 130, 24, 23, 193, 102, 174, 175, 65, 91, 53, 192, 38, 92, 163, 10, 59, 186, 66, 182, 58, 197, 129, 13, 21, 30, 128, 194, 46, 205]
Multi-objective	Effectiveness	[199, 215, 216, 139, 181, 169, 67, 212, 167, 134, 9, 143, 157, 156, 158, 137, 170, 146, 11, 112, 111, 108, 140, 81, 120, 123, 151, 152, 100, 180, 200, 88, 104, 56, 37, 121, 178, 98, 154, 34, 166, 86, 27, 125, 25, 135, 22, 79, 189, 43, 87, 211, 171, 187, 15, 191, 162, 60, 160, 159, 16, 38, 52, 50, 116, 5, 198, 77, 188, 196, 32, 185, 28, 44, 42, 122, 7, 13, 2, 12, 40, 179, 17, 97]
	Efficiency	[167, 140]
	Diversity	[157, 156, 111, 108, 112, 143, 81, 120, 151, 152, 104, 56, 34, 86, 125, 25, 187, 191, 160, 159, 16, 38, 198, 77, 188, 32, 13, 17, 97]
	Complexity	[169, 212, 143, 123, 180, 200, 37, 166, 27, 79, 87, 211, 162, 5, 185, 28, 122, 12, 139, 158, 100, 56, 98, 154, 189, 38, 188, 171, 7]

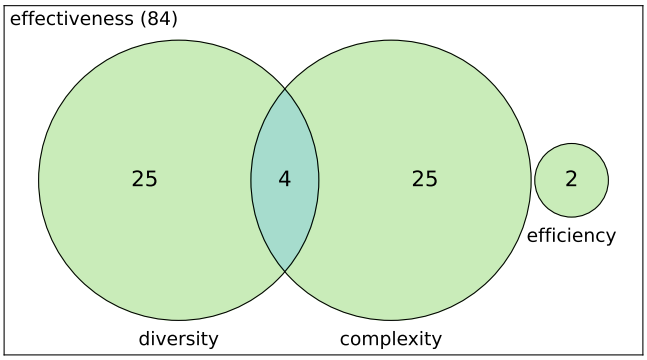

Figure 7: Distribution of objectives across EAs using multiple objectives. Effectiveness (predictive performance) is optimized in all 84 studies. From these, 56 work optimize another objective, be it diversity, complexity, or efficiency. Only four studies [188, 56, 143, 38] optimize three objectives (effectiveness, diversity and complexity), and no study optimizes all four at the same time.

8 Types of Evolutionary Algorithms

A variety of EAs have been employed in ensemble learning. While some of these EAs are less frequent in literature (e.g. Flower Pollination Algorithm [213], Levy-Flight Firefly Algorithm [212]), others are more common. Among them, Genetic Algorithms (GAs) is the most popular, followed by Genetic Programming, and Differential Evolution.

Within GAs, apart from its *vanilla* version, Non-dominated Sorting Genetic Algorithm II (NSGA-II) is the most popular implementation. This choice seems due to NSGA-II's ability to deal with multiple objectives, suiting well the multi-objective nature of ensemble learning. Table 8 shows the distribution of the surveyed studies according to the type of EA used.

9 Types of Base Learners

In the surveyed studies, the most commonly used type of base learner is artificial neural networks, used in 75 studies; followed by tree-based algorithms (e.g. decision trees, arithmetic trees), used in 48 studies; and support vector machines, used in 46 studies. The number of studies using each type of base learner algorithm is shown in Table 9.

Some ensemble techniques are more appropriate for some type(s) of selected base learner(s). Support Vector Machines, for instance, are stable classifiers, making the techniques of selecting either instances or attributes for each base learner inefficient as a diversity inductor [13].

The majority of the studies use homogeneous ensembles, as opposed to heterogeneous ones (113 vs 47). Four work use both types of ensembles, which brings the count to 117 and 51, respectively. Figure 8 shows the base learners that were used in at least 5 studies, as well as the study's ensemble type: either homogeneous, or heterogeneous.

Homogeneous ensembles are composed by the same base learner paradigm. However, this is not to say that all base learners are exactly the same. When using neural networks, those models can have distinct activation functions or topologies. According to Rahman and Verma [161], there are five strategies for inducing diverse models in homogeneous ensembles: (i) post-model optimization (see Section 4.3); (ii) manipulation of the error function; (iii) distinct attribute subsets across base learners (see Section 4.2); (iv) manipulation of output targets, in which some instances in the training set have their class labels switched, for inducing diversity; and (v) distinct instance subsets across base learners (see Section 4.1). Strategies (ii) and (iv) are not covered in this survey, though, due to the lack of relevant papers.

Table 8 Studies organized by the type of EA employed, and the ensemble learning stage optimized.

Evolutionary Family	Generation	Selection	Integration
Flower Pollination			[213]
Clonal Selection	[13]	[13]	
Evolutionary Algorithm	[163]		
Inclined Planes Optimization		[158]	[158]
Multi-Objective EA		[12, 11]	[11]
Moth-Flame Optimization			[212]
Levy-flight firefly Algorithm			[212]
Virus-Evolutionary Genetic Algorithm			[76]
Many-Objectives Evolutionary Algorithm		[7]	
Evolutionary Strategy	[197, 205]		
Artificial Bee Colony	[24, 23]	[152, 175, 151]	[91, 175]
Estimation of Distribution Algorithm	[63, 29, 62, 25, 61, 64]	[29, 25]	[63, 21, 62, 61, 64]
Particle Swarm Optimization	[30, 157, 35, 29, 102, 156, 36, 94, 150, 37, 93, 13]	[55, 29, 158, 13]	[170, 158, 4]
Differential Evolution	[196, 177, 43, 44, 176, 52, 193, 53, 181, 42, 114, 180, 178, 179, 123]	[46, 181, 123]	[177, 215, 216, 146, 214, 83, 176, 180, 178, 179, 26]
Genetic Programming	[16, 119, 69, 17, 30, 164, 58, 79, 199, 130, 183, 132, 71, 133, 70, 68, 190, 194, 120, 47]	[58]	[119, 124, 191, 192, 127, 60, 4, 120, 72]
Genetic Algorithms	[6, 28, 99, 119, 66, 10, 116, 58, 135, 134, 187, 169, 9, 211, 95, 5, 185, 143, 122, 112, 111, 92, 129, 171, 162, 81, 57, 108, 52, 174, 190, 101, 167, 200, 166, 96, 77, 121, 140, 33, 67, 40, 196, 125, 87, 41, 2, 13]	[97, 86, 6, 55, 139, 116, 58, 59, 46, 169, 153, 32, 100, 131, 122, 160, 12, 90, 98, 198, 89, 186, 56, 166, 27, 188, 140, 104, 38, 159, 34, 11, 154, 137, 13]	[109, 107, 119, 189, 22, 139, 100, 143, 112, 111, 90, 88, 172, 98, 15, 89, 108, 65, 128, 77, 206, 110, 141, 142, 38, 125, 41, 11]

Table 9 Studies organized according to the base learners they employ. We only show in this table base learners that are present in at least 5 papers. For the complete list of base learners, please refer to our website at `https://henryzord.github.io/eael`.

Base Learner	Related work
Gaussian Process Regression	[200, 142, 141, 15, 129]
Linear Regression	[89, 142, 90, 141, 121]
Fuzzy rule-based Classifier	[189, 38, 135, 87, 66, 188]
Conditional Random Fields	[177, 176, 65, 180, 179, 178, 181]
Logistic Regression	[146, 60, 72, 83, 169, 86, 175]
Random Forest	[169, 137, 4, 166, 60, 200, 193, 86]
Rule-based	[11, 181, 133, 132, 171, 41, 50, 52, 135, 164, 12]
Naïve Bayes	[137, 4, 116, 83, 72, 214, 146, 60, 211, 157, 156, 175, 88, 95, 86, 40]
K-Nearest Neighbor	[175, 72, 60, 158, 157, 156, 114, 140, 200, 4, 88, 90, 55, 56, 104, 154, 153, 196, 100, 83, 86, 170, 214, 125, 97, 98, 137, 7, 40]
Support Vector Machines	[81, 169, 156, 167, 107, 152, 93, 4, 166, 170, 205, 176, 97, 7, 100, 177, 86, 13, 191, 181, 94, 141, 154, 160, 159, 60, 142, 162, 153, 192, 129, 40, 193, 90, 98, 89, 212, 83, 200, 146, 65, 22, 157, 26, 34, 125]
Trees	[95, 169, 9, 214, 16, 190, 21, 172, 10, 166, 137, 7, 194, 71, 108, 111, 86, 72, 185, 69, 47, 28, 130, 183, 112, 68, 134, 40, 199, 110, 120, 175, 127, 70, 83, 79, 197, 200, 121, 17, 58, 119, 140, 5, 77, 109, 124, 125]
Artificial Neural Network	[64, 156, 53, 25, 215, 174, 32, 10, 166, 187, 170, 6, 96, 101, 198, 67, 37, 97, 7, 92, 100, 122, 46, 206, 86, 158, 30, 143, 29, 23, 36, 191, 185, 28, 102, 130, 141, 154, 42, 33, 153, 142, 60, 27, 192, 62, 139, 129, 40, 120, 90, 61, 98, 43, 99, 89, 212, 63, 83, 216, 200, 128, 123, 213, 59, 65, 44, 186, 24, 182, 157, 57, 35, 150, 163]

Heterogeneous ensembles comprise base learners from distinct paradigms. As such, there is no pressure for inducing diversity in the ensemble, since learners from distinct paradigms tend to make diverse predictions. In our survey, among the studies using diversity measures, 19 have a homogeneous set of base learners, while 9 use a heterogeneous set. These numbers include a single paper that proposes both homogeneous and heterogeneous ensembles. The larger number

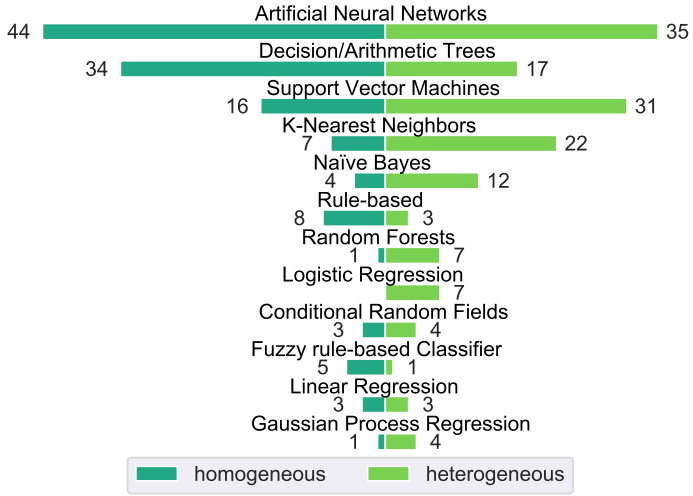

Figure 8: Types of base learners used in surveyed work, as well as their distribution.

of papers with homogeneous sets is probably because it is simpler to work with homogeneous ensembles than with heterogeneous ones. While Figure 8 depicts an overview of the number of studies per type of base learners, Table 10 identifies which studies are using which types of base learners, and also their configuration (homogeneous or heterogeneous). Note that a few studies (4) use both types of ensemble.

Table 10 Studies organized according to the base learners' homogeneity/heterogeneity.

Homogeneity	Related work
Homogeneous	[134, 143, 91, 167, 6, 199, 215, 205, 112, 66, 216, 67, 196, 139, 26, 182, 96, 140, 81, 41, 119, 152, 180, 95, 127, 111, 133, 131, 164, 22, 33, 79, 32, 190, 90, 47, 76, 64, 110, 88, 107, 189, 58, 43, 44, 124, 108, 24, 211, 132, 171, 128, 193, 179, 178, 187, 102, 174, 162, 15, 172, 109, 160, 42, 46, 159, 52, 50, 62, 77, 188, 63, 61, 37, 38, 198, 5, 23, 53, 116, 92, 194, 17, 121, 36, 150, 35, 183, 94, 16, 93, 9, 71, 206, 29, 55, 99, 56, 69, 30, 70, 27, 57, 104, 68, 87, 101, 197, 122, 13, 2, 12, 21]
Heterogeneous	[157, 213, 83, 158, 137, 170, 156, 146, 11, 72, 212, 169, 181, 114, 123, 151, 86, 100, 200, 4, 135, 142, 214, 130, 141, 166, 191, 175, 60, 65, 59, 10, 177, 186, 163, 192, 176, 25, 98, 125, 97, 154, 153, 34, 129, 7, 40]
Both	[120, 89, 185, 28]

10 Algorithm Complexity by Stages of Ensemble Learning

As it is expected with a survey that broadly reviews the literature, it is difficult to derive a single algorithm complexity for Ensemble Learning with Evolutionary Algorithms, or even multiple accurate estimations. For this reason, in this section we will discuss what aspects have the most impact on algorithm complexity, as well as deriving a general complexity for relevant ensemble stages (i.e. generation and selection).

The most time-consuming step in an evolutionary algorithm is evaluating candidate solutions. How candidate solutions are built differs among EAs, but it is safe to assume that these solutions are, in the context of ensemble learning, already-built, ready-to-deploy ensembles. Consider for example a "generic" evolutionary algorithm for the generation stage. Let us assume that this EA does not have a pool of base classifiers – that is, for each candidate solution, it builds base classifiers that will only be used by that solution. If this EA runs for G generations and has a population of S individuals, then the time complexity of this algorithm is $O(G \times (BS\Psi\Omega))$, where B is the number of base classifiers in each ensemble (individual), Ψ is the complexity of the most time-consuming base classifier employed in the ensemble (in case it is heterogeneous), or the complexity of the only base classifier used (in case the ensemble is homogeneous), and Ω is

the complexity of aggregating predictions among base classifiers. Ω can be as fast as $O(B)$, when using majority voting, to an arbitrarily complex algorithm, such as employing Genetic Algorithms to evolve Expression Trees (presented in Section 6.2.1). When more than one aggregation policy is available, the reader should assume the worst case scenario – that is, that all solutions will take as much time to train as it takes to use the most time-consuming base classifiers, coupled with the most time-consuming aggregation policy.

Note that having a pre-built pool of classifiers to choose from reduces the time complexity. In the case of a "generic" EA for the selection stage, when the EA performs static selection, the time complexity is $O((P \times \Psi) + G \times (S\Omega))$, where P is the pool size, and $P \geq B$ (in this case, each individual will have a different B). Once base classifiers are built, their matrices of probabilities can be stored in memory and aggregated by any desired aggregation policy with Ω complexity. Since static selection consists only in flipping bits in a binary vector, its complexity is negligible. On the other hand, the complexity of dynamic selection (presented in Section 5.2) lies on the complexity of training a selector to be later used during the prediction phase. In this case, the complexity of a generic EA performing dynamic selection is approximately $O((P \times \Psi) + G \times (S\Xi\Omega))$, where Ξ is the complexity of learning that selector.

11 Conclusions and New Research Directions

Ensemble learning is an extensive research field due to the improvement it presents in comparison to single learners and the easiness to integrate within some challenging types of machine learning problems (e.g. data stream learning and datasets with imbalanced class distributions [134, 112, 72]). For some problems, inducing a single, stronger-than-all base learner can be a difficult task [6]; whilst ensembles of models can perform better with regard to both effectiveness and efficiency [114, 81, 18, 140, 151].

Ensemble learning can be further enhanced by using EAs in one or more of its learning stages: generation, selection, and integration. In this survey, we reviewed a large number of studies using many types of EAs for ensemble learning, and proposed a taxonomy to classify such studies with regard to different aspects of ensemble learning. We also reviewed the debate on controversial topics, like the selection of ensemble members (rather than using all members) and the usefulness of optimizing a diversity measure for the members of the ensemble.

In order to facilitate the review of specific studies discussed in our survey, we make available the metadata used to compile our figures and tables. By using such metadata, one can see at a glance all the main aspects of a given study (e.g. which base learners, objectives, learning stages, etc, a study is using). The repository is available at `https://github.com/henryzord/eael`. We also provide a master table, in the form of a website, listing all surveyed work, and their classification according to our taxonomy: `https://henryzord.github.io/eael`.

11.1 Summary of Findings

First, we discuss the main findings of this survey regarding each stage of the ensemble learning process. The generation stage, i.e. where the ensemble members are generated, was found to be the most popular step to employ EAs, having more studies dedicated to it than the selection and integration stages combined. Wrapper methods were found to be much more common than filter ones for the instance selection and attribute selection approaches. This seems natural, considering that, unlike filters, wrappers select attributes or instances customized for the supervised learning algorithm to be used later (to induce a model), which tends to improve predictive performance. However, wrappers are normally much slower than filters. Hence, in applications with large datasets or where efficiency is a critical factor, the filter approach deserves more attention. In addition, in the model tuning approach for generation, post-model optimization (used to improve an existing model) was found to be more popular than pre-model optimization (used before learning the model).

The next stage, selection – where ensemble members are selected to be used in the testing phase – is an optional stage, which is missing in many ensemble learning systems. In this stage, static selection, where the regions of competence of ensemble members are identified at training time, was found to be much more popular than dynamic selection, where those regions are identified at testing (prediction) time. This seems partly due to the greater simplicity and computational efficiency of the former, since dynamic selection in general requires a more time-consuming process of identifying regions of competence of ensemble members for each testing instance.

In the integration stage, where the predictions of the base learners are integrated into a final prediction for each instance, by far the most popular approach among the surveyed studies was the use of a first degree polynomial – a simple linear approach. Among the non-linear integration techniques, the most common was the use of expression trees, using a genetic programming algorithm. It seems that more research is needed on non-linear techniques for integration, in order to determine whether or not their higher computational complexity could be justified by a significant increase in predictive performance.

Regarding the number of objectives in the fitness function, multi-objective EAs were found to be much more common than single-objective ones. This seems natural, given the multi-objective nature of the ensemble learning problem. In terms of specific types of objectives, effectiveness (predictive performance) is used by all surveyed EAs, since it is essential. Diversity and complexity share a second place, despite diversity being a controversial objective, as discussed earlier. Finally, efficiency, the capacity to generate ensembles that are computationally fast, is optimized in only two studies. Efficiency is important in huge datasets (since the training process must eventually finish, and a compromise between time spent in training stage and effectiveness must be made), and in data stream scenarios, where data is treated not as a fixed batch of instances, but instead as a continuous flow. Efficiency and effectiveness are competing objectives, since efficiency prioritizes models that are faster to train (and thus more likely to be simpler, less accurate models). However, if efficiency is a priority, one could use techniques such as parallelization of one of evolutionary algorithms' steps [85] to alleviate this competition.

Regarding the main types of EAs used in the surveyed studies, the most popular one was by far Genetic Algorithms (often NSGA-II, a multi-objective GA), followed by Genetic Programming and Differential Evolution.

Regarding the main type of base learner, the most popular one was Artificial Neural Networks (ANNs), followed by decision trees and Support Vector Machines (SVMs). The popularity of ANNs as base learners seems partly due to a long history of interaction in the EA and ANN research areas, and partly due to the nature of ANNs, whose performance can often be improved when using ensembles. However, learning ensembles of ANNs or SVMs tends to be very computationally expensive. This problem is mitigated when learning an ensemble of decision trees (much faster base learners).

11.2 New Research Directions

One direction for future research is the automated selection of the best combination of ensemble algorithms and their hyper-parameter settings for a given input dataset. This is a complex optimization problem because, as discussed earlier, there are many types of ensembles (e.g. bagging, boosting, stacking, etc), and for each type of ensemble, many types of base (classification or regression) algorithms can be chosen. In addition, both the ensemble type and its base algorithm type(s) typically have many hyper-parameters, whose settings also have a large influence on the ensemble's predictive performance. All these choices of algorithms and hyper-parameter settings interact in a complex manner, and ideally all these choices should be made in a synergistic way, optimizing all these choices as a whole for the specific dataset provided as input by the user. Emerging research has addressed this complex optimization problem by doing a search in the space of different types of learning algorithms and their hyper-parameter settings, in order to automatically select the best combination of algorithm and hyper-parameter

settings for an input dataset [105],[201]. To the best of our knowledge, although there are several EA-based systems that address this problem by considering a search space with many types of supervised learning algorithms (e.g. [45],[144]), there are only two studies using an EA to address this problem by considering a search space focused on ensembles [105],[207]. This seems an area with good potential for research growth.

Also, it is yet to be seen a framework that provides a synergistic integration of two or more ensemble learning stages (generation, selection and integration). Even when a study addresses two or more stages, this is not done synergistically; base learners are first generated and later selected, or first selected and later integrated. We are not aware of any EA addressing all three stages.

Finally, ensemble learning with EAs would benefit from a unified, generic-purpose software tool, similarly to what WEKA [202] and scikit-learn [155] do for machine learning in general and Tensorflow [1] for deep learning. This would greatly facilitate the task of comparing different strategies, e.g., distinct approaches for generating or selecting base learners while keeping the same fitness function. We believe the development of such a framework to be a major step forward to the evolutionary ensemble learning community.

11.3 Further Readings

While this work is, up to our knowledge, the first to present a broad review of evolutionary algorithms for ensemble learning, the following literature can give a better understanding on topics related to ensemble learning, not necessarily involving evolutionary algorithms.

A closely-related work to ours is the one of Yao and Islam [210], which present a review of evolutionary algorithms for ensemble learning, although focusing only in work that uses artificial neural networks as base classifiers.

For a general comprehension on ensemble learning, not necessarily involving evolutionary algorithms, Sagi and Rokach [168] presents recent, state-of-the-art methods for ensemble learning. This is an update of another review on ensemble learning of the same author, presented in the work of Rokach [165]. While the former reviews generation and integration methods, as well as presenting the main challenges when building methods for ensemble learning, the later reviews integration and selection methods, and a discussion on ensemble diversity.

In the work of Oza and Tumer [149] the authors review ensemble methods for solving real-world problems, such as remote sensing, person recognition, and medicine applications. Athar et al. [8] review classifier ensembles for sentiment analysis. Data stream analysis with ensembles is reviewed both in the work of Gomes et al. [80] and Krawczyk et al. [113].

A broad review on classification ensembles and its applications is presented in [184], while a survey on regression ensembles is presented in [136].

Regarding the generation stage, Olvera-Lopez et al. [145] presents a review on instance selection methods, which can be used in this stage of ensemble learning; while Debie et al. [51] review ensemble methods that focus on feature selection.

For the selection stage, Britto et al. [20] presents a review on dynamic selection of classifiers, as well as a statistical comparison of results of the reviewed methods. An update of the reviewed methods, as well as the proposed taxonomy by the authors, is presented by Cruz et al. [39].

Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. The authors would also like to acknowledge FAPERGS for partially funding this research.

Declaration of Interests

Competing interests: The authors declare none.

References

[1] M. Abadi et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. URL `https://www.tensorflow.org/`.

[2] J. Adair, A. Brownlee, F. Daolio, and G. Ochoa. Evolving Training sets for Improved Transfer Learning in Brain Computer Interfaces. In *International Workshop on Machine Learning, Optimization, and Big Data*, pages 186–197. Springer, 2017.

[3] W. A. Albukhanajer, Y. Jin, and J. A. Briffa. Classifier Ensembles for Image Identification using Multi-objective Pareto Features. *Neurocomputing*, 238:316–327, May 2017.

[4] S. Ali and A. Majid. Can–Evo–Ens: Classifier Stacking based Evolutionary Ensemble System for Prediction of Human Breast Cancer using Amino Acid Sequences. *Journal of Biomedical Informatics*, 54:256–269, April 2015.

[5] M. S. Aliakbarian and A. Fanian. Internet Traffic Classification Using MOEA and Online Refinement in Voting on Ensemble Methods. In *Iranian Conference on Electrical Engineering*, pages 1–6. IEEE, 2013.

[6] L. M. Almeida and P. S. Galvão. Ensembles with Clustering-and-Selection Model Using Evolutionary Algorithms. In *Brazilian Conference on Intelligent Systems*, pages 444–449. IEEE, 2016.

[7] M. Asafuddoula, B. Verma, and M. Zhang. A Divide-and-Conquer Based Ensemble Classifier Learning by Means of Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation*, 22(5), December 2017.

[8] A. Athar et al. Exploring the Ensemble of Classifiers for Sentimental Analysis: A Systematic Literature Review. In *International Conference on Machine Learning and Computing*, pages 410–414. ACM, 2017.

[9] D. A. Augusto, H. J. C. Barbosa, and N. F. F. Ebecken. Coevolutionary Multi-population Genetic Programming for Data Classification. In *Conference on Genetic and Evolutionary Computation*, pages 933–940. ACM, 2010.

[10] M. A. Bagheri, Q. Gao, and S. Escalera. A Genetic-based Subspace Analysis method for Improving Error-correcting Output Coding. *Pattern Recognition*, 46(10):2830–2839, October 2013.

[11] V. Basto-Fernandes et al. A Spam Filtering Multi-objective Optimization Study Covering Parsimony Maximization and Three-way Classification. *Applied Soft Computing*, 48:111–123, November 2016.

[12] V. Basto-Fernandes et al. Quadcriteria Optimization of Binary Classifiers: Error Rates, Coverage, and Complexity. In A.-A. Tantar, E. Tantar, M. Emmerich, P. Legrand, L. Alboaie, and H. Luchian, editors, *EVOLVE – A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, pages 37–49. Springer, 2018.

[13] J. d. O. Batista, R. B. Rodrigues, and F. M. Varejão. Soft Computing Classifier Ensemble for Fault Diagnosis. In *International Symposium on Industrial Electronics*, pages 1348–1353. IEEE, 2017.

[14] M. Á. Bautista, O. Pujol, X. Baró, and S. Escalera. Introducing the Separability Matrix for Error Correcting Output Codes Coding. In *International Workshop on Multiple Classifier Systems*, pages 227–236. Springer, 2011.

[15] Y. Bazi et al. Robust Estimation of Water Chlorophyll Concentrations with Gaussian Process Regression and IOWA Aggregation Operators. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(7):3019–3028, June 2014.

[16] U. Bhowan, M. Johnston, and M. Zhang. Evolving Ensembles in Multi-objective Genetic Programming for Classification with Unbalanced Data. In *Conference on Genetic and Evolutionary Computation*, pages 1331–1338. ACM, 2011.

[17] U. Bhowan, M. Johnston, and M. Zhang. Comparing Ensemble Learning Approaches in Genetic Programming for Classification with Unbalanced Data. In *Conference on Genetic and Evolutionary Computation*, pages 135–136. ACM, 2013.

[18] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, August 1996.

[19] L. Breiman. Random Forests. *Machine learning*, 45(1):5–32, October 2001.

[20] A. S. Britto Jr., R. Sabourin, and L. E. S. Oliveira. Dynamic Selection of Classifiers – A Comprehensive Review. *Pattern Recognition*, 47(11):3665–3680, November 2014.

[21] H. Cagnini, M. Basgalupp, and R. Barros. Increasing Boosting Effectiveness with Estimation of Distribution Algorithms. In *Congress on Evolutionary Computation*, pages 1–8. IEEE, 2018.

[22] J.-J. Cao, S. Kwong, R. Wang, and K. Li. An Indicator-based Selection Multi-objective Evolutionary Algorithm with Preference for Multi-class Ensemble. In *International Conference on Machine Learning and Cybernetics*, pages 147–152. IEEE, 2014.

[23] P. Cao, B. Li, D. Zhao, and O. Zaiane. A Novel Cost Sensitive Neural Network Ensemble for Multiclass Imbalance Data Learning. In *International Joint Conference on Neural Networks*, pages 1–8. IEEE, 2013.

[24] P. Cao, D. Zhao, and O. Zaiane. Measure Optimized Cost-sensitive Neural Network Ensemble for Multiclass Imbalance Data Learning. In *International Conference on Hybrid Intelligent Systems*, pages 35–40. IEEE, 2013.

[25] P. A. D. Castro and F. J. Von Zuben. Learning Ensembles of Neural Networks by Means of a Bayesian Artificial Immune System. *IEEE Transactions on Neural Networks*, 22(2): 304–316, February 2011.

[26] R. K. Chaurasiya, N. D. Londhe, and S. Ghosh. Binary DE-based Channel Selection and Weighted Ensemble of SVM Classification for Novel Brain–computer Interface using Devanagari script-based P300 Speller Paradigm. *International Journal of Human–Computer Interaction*, 32(11):861–877, July 2016.

[27] H. Chen and X. Yao. Evolutionary Multiobjective Ensemble Learning based on Bayesian Feature Selection. In *Congress on Evolutionary Computation*, pages 267–274. IEEE, 2006.

[28] W.-C. Chen, L.-Y. Tseng, and C.-S. Wu. A Unified Evolutionary Training scheme for Single and Ensemble of Feedforward Neural Network. *Neurocomputing*, 143:347–361, November 2014.

[29] Y. Chen and Y. Zhao. A novel Ensemble of Classifiers for Microarray Data Classification. *Applied Soft Computing*, 8(4):1664–1669, September 2008.

[30] Y. Chen, B. Yang, and A. Abraham. Flexible Neural Trees Ensemble for Stock Index Modeling. *Neurocomputing*, 70(4):697–703, January 2007.

[31] C.-Y. Chiu and B. Verma. Effect of Varying Hidden Neurons and Data Size on Clusters, Layers, Diversity and Accuracy in Neural Ensemble Classifier. In *International Conference on Computational Science and Engineering*, pages 455–459. IEEE, 2013.

[32] C.-Y. Chiu and B. Verma. Multi-objective Evolutionary Algorithm Based Optimization of Neural Network Ensemble Classifier. In *International Conference on Signal Processing and Communication Systems*, pages 1–5. IEEE, 2014.

[33] D. Chyzhyk, A. Savio, and M. Graña. Computer Aided Diagnosis of Schizophrenia on Resting State fMRI data by Ensembles of ELM. *Neural Networks*, 68:23–33, August 2015.

[34] A. L. V. Coelho, C. A. M. Lima, and F. J. Von Zuben. GA-based Selection of Components for Heterogeneous Ensembles of Support Vector Machines. In *Congress on Evolutionary Computation*, pages 2238–2245. IEEE, 2003.

[35] J.-F. Connolly, E. Granger, and R. Sabourin. Comparing Dynamic PSO Algorithms for Adapting Classifier Ensembles in Video-based Face Recognition. In *Workshop on Computational Intelligence in Biometrics and Identity Management*, pages 1–8. IEEE, 2011.

[36] J.-F. Connolly, E. Granger, and R. Sabourin. Evolution of Heterogeneous Ensembles through Dynamic Particle Swarm Optimization for Video-based Face Recognition. *Pattern Recognition*, 45(7):2460–2477, July 2012.

[37] J.-F. Connolly, E. Granger, and R. Sabourin. Dynamic Multi-objective Evolution of Classifier Ensembles for Video Face Recognition. *Applied Soft Computing*, 13(6):3149–3166, June 2013.

[38] O. Cordón and K. Trawiński. A Novel Framework to Design Fuzzy Rule-based Ensembles using Diversity Induction and Evolutionary Algorithms-based Classifier Selection and Fusion. In *International Work-Conference on Artificial Neural Networks*, pages 36–58. Springer, 2013.

[39] R. M. O. Cruz, R. Sabourin, and G. D. C. Cavalcanti. Dynamic Classifier Selection: Recent Advances and Perspectives. *Information Fusion*, 41:195–216, May 2018.

[40] A. K. Das, S. Das, and A. Ghosh. Ensemble Feature Selection using Bi-objective Genetic Algorithm. *Knowledge-Based Systems*, 123:116–127, May 2017.

[41] S. A. Davidsen and M. Padmavathamma. Multi-modal Evolutionary Ensemble Classification in Medical Diagnosis Problems. In *International Conference on Advances in Computing, Communications and Informatics*, pages 1366–1370. IEEE, 2015.

[42] T. P. F. De Lima and T. B. Ludermir. Optimizing Dynamic Ensemble Selection Procedure by Evolutionary Extreme Learning Machines and a Noise Reduction Filter. In *International Conference on Tools with Artificial Intelligence*, pages 546–552. IEEE, 2013.

[43] T. P. F. de Lima and T. B. Ludermir. Ensembles of Evolutionary Extreme Learning Machines through Differential Evolution and Fitness Sharing. In *International Joint Conference on Neural Networks*, pages 2677–2682. IEEE, 2014.

[44] T. P. F. de Lima, A. T. Sergio, and T. B. Ludermir. Improving Classifiers and Regions of Competence in Dynamic Ensemble Selection. In *Brazilian Conference on Intelligent Systems*, pages 13–18. IEEE, 2014.

[45] A. G. C. de Sá, W. J. G. S. Pinto, L. O. V. B. Oliveira, and G. L. Pappa. RECIPE: a Grammar-based Framework for Automatically Evolving Classification Pipelines. In *European Conference on Genetic Programming*, pages 246–261. Springer, 2017.

[46] C. De Stefano, A. D. Cioppa, and A. Marcelli. Evolutionary Approaches for Pooling Classifier Ensembles: Performance Evaluation. In *International Conference of Soft Computing and Pattern Recognition*, pages 309–314. IEEE, 2013.

[47] C. De Stefano, G. Folino, F. Fontanella, and A. S. Di Freca. Using Bayesian Networks for Selecting Classifiers in GP Ensembles. *Information Sciences*, 258:200–216, February 2014.

[48] K. Deb. Multi-objective optimisation using evolutionary algorithms: an introduction. In *Multi-objective evolutionary optimisation for product design and manufacturing*, pages 3–34. Springer, 2011.

[49] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[50] E. Debie, K. Shafi, C. Lokan, and K. Merrick. Performance Analysis of Rough set Ensemble of Learning Classifier Systems with Differential Evolution based rule Discovery. *Evolutionary Intelligence*, 6(2):109–126, October 2013.

[51] E. Debie, K. Shafi, K. Merrick, and C. Lokan. On Taxonomy and Evaluation of Feature Selection-Based Learning Classifier System Ensemble Approaches for Data Mining Problems. *Computational Intelligence*, 33(3):554–578, July 2016.

[52] E. S. Debie, K. Shafi, and C. Lokan. REUCS-CRG: Reduct based Ensemble of Supervised Classifier System with Combinatorial Rule Generation for Data Mining. In *Conference on Genetic and Evolutionary Computation*, pages 1251–1258. ACM, 2013.

[53] S. Dehuri, A. K. Jagadev, and S.-B. Cho. Epileptic Seizure Identification from Electroencephalography signal using DE-RBFNs Ensemble. *Procedia Computer Science*, 23:84–95, November 2013.

[54] T. G. Dietterich. Ensemble Methods in Machine Learning. In *International Workshop on Multiple Classifier Systems*, pages 1–15. Springer, 2000.

[55] E. M. Dos Santos, L. S. Oliveira, R. Sabourin, and P. Maupin. Overfitting in the Selection of Classifier Ensembles: a Comparative study Between PSO and GA. In *Conference on Genetic and Evolutionary Computation*, pages 1423–1424. ACM, 2008.

[56] E. M. Dos Santos, R. Sabourin, and P. Maupin. Pareto Analysis for the Selection of Classifier Ensembles. In *Conference on Genetic and Evolutionary Computation*, pages 681–688. ACM, 2008.

[57] P. Duell, I. Fermin, and X. Yao. Speciation Techniques in Evolved Ensembles with Negative Correlation Learning. In *Congress on Evolutionary Computation*, pages 3317–3321. IEEE, 2006.

[58] E. Dufourq and N. Pillay. Hybridizing Evolutionary Algorithms for Creating Classifier Ensembles. In *World Congress on Nature and Biologically Inspired Computing*, pages 84–90. IEEE, 2014.

[59] E. J. d. R. e Silva, T. B. Ludermir, and L. M. Almeida. Clustering and Selection using Grouping Genetic Algorithms for Blockmodeling to construct Neural Network Ensembles. In *International Conference on Tools with Artificial Intelligence*, pages 420–425. IEEE, 2013.

[60] H. J. Escalante, N. Acosta-Mendoza, A. Morales-Reyes, and A. Gago-Alonso. Genetic Programming of Heterogeneous Ensembles for Classification. In *Iberoamerican Congress on Pattern Recognition*, pages 9–16. Springer, 2013.

[61] T. Escovedo, A. da Cruz, M. Vellasco, and A. Koshiyama. NEVE: A Neuro-evolutionary Ensemble for Adaptive Learning. In *International Conference on Artificial Intelligence Applications and Innovations*, pages 636–645. Springer, 2013.

[62] T. Escovedo, A. V. A. da Cruz, M. Vellasco, and A. S. Koshiyama. Using Ensembles for Adaptive Learning: A Comparative Approach. In *International Joint Conference on Neural Networks*, pages 1–7. IEEE, 2013.

[63] T. Escovedo, A. V. A. da Cruz, M. M. Vellasco, and A. S. Koshiyama. Learning Under Concept Drift Using a Neuro-evolutionary Ensemble. *International Journal of Computational Intelligence and Applications*, 12(4):1340002, December 2013.

[64] T. Escovedo et al. NEVE++: A Neuro-evolutionary Unlimited Ensemble for Adaptive Learning. In *International Joint Conference on Neural Networks*, pages 3331–3338. IEEE, 2014.

[65] I. Fatima, M. Fahim, Y.-K. Lee, and S. Lee. Classifier Ensemble Optimization for Human Activity Recognition in Smart Homes. In *International Conference on Ubiquitous Information Management and Communication*, pages 1 – 7. ACM, 2013.

[66] A. Fernández, S. del Río, and F. Herrera. A First Approach in Evolutionary Fuzzy Systems based on the Lateral Tuning of the Linguistic Labels for Big Data Classification. In *International Conference on Fuzzy Systems*, pages 1437–1444. IEEE, 2016.

[67] J. C. Fernández, M. Cruz-Ramírez, and C. Hervás-Martínez. Sensitivity versus Accuracy in Ensemble Models of Artificial Neural Networks from Multi-objective Evolutionary Algorithms. *Neural Computing and Applications*, 30(1):289–305, December 2016.

[68] G. Folino, C. Pizzuti, and G. Spezzano. Improving Cooperative GP Ensemble with Clustering and Pruning for Pattern Classification. In *Conference on Genetic and Evolutionary Computation*, pages 791–798. ACM, 2006.

[69] G. Folino, C. Pizzuti, and G. Spezzano. An Adaptive Distributed Ensemble Approach to Mine Concept-drifting Data Streams. In *International Conference on Tools with Artificial Intelligence*, pages 183–188. IEEE, 2007.

[70] G. Folino, C. Pizzuti, and G. Spezzano. StreamGP: Tracking Evolving GP Ensembles in Distributed Data Streams using Fractal Dimension. In *Conference on Genetic and Evolutionary Computation*, pages 1751–1751. ACM, 2007.

[71] G. Folino, C. Pizzuti, and G. Spezzano. An Ensemble-based Evolutionary Framework for Coping with Distributed Intrusion Detection. *Genetic Programming and Evolvable Machines*, 11(2):131–146, February 2010.

[72] G. Folino, F. S. Pisani, and P. Sabatino. An Incremental Ensemble Evolved by using Genetic Programming to Efficiently Detect drifts in Cyber Security Datasets. In *Conference on Genetic and Evolutionary Computation*, pages 1103–1110. ACM, 2016.

[73] A. A. Freitas. A Critical Review of Multi-objective Optimization in Data Mining: A Position Paper. *ACM SIGKDD Explorations Newsletter*, 6(2):77–86, December 2004.

[74] Y. Freund and R. E. Schapire. A Desicion-theoretic Generalization of on-line Learning and an Application to Boosting. In *European Conference on Computational Learning Theory*, pages 23–37. Springer, 1995.

[75] Y. Freund and R. E. Schapire. Experiments with a new Boosting Algorithm. In *International Conference on Machine Learning*, pages 148–156. International Machine Learning Society, 1996.

[76] D. Fuqiang, Z. Minqing, and L. Jia. Virus-Evolutionary Genetic Algorithm Based Selective Ensemble for Steganalysis. In *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 553–558. IEEE, 2014.

[77] M. Galar, A. Fernández, E. Barrenechea, and F. Herrera. EUSBoost: Enhancing Ensembles for Highly Imbalanced Data-sets by Evolutionary Undersampling. *Pattern Recognition*, 46 (12):3460–3471, December 2013.

[78] M. Galea, Q. Shen, and J. Levine. Evolutionary Approaches to Fuzzy Modelling for Classification. *The Knowledge Engineering Review*, 19(1):27–59, April 2004.

[79] A. Garg and J. S. L. Lam. Improving Environmental Sustainability by Formulation of Generalized Power Consumption Models using an Ensemble based Multi-gene Genetic Programming Approach. *Journal of Cleaner Production*, 102:246–263, September 2015.

[80] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet. A Survey on Ensemble Learning for Data Stream Classification. *ACM Computing Surveys*, 50(2):23, March 2017.

[81] S. Gu and Y. Jin. Generating Diverse and Accurate Classifier Ensembles using Multi-objective Optimization. In *Symposium on Computational Intelligence in Multi-Criteria Decision-Making*, pages 9–15. IEEE, 2014.

[82] L. K. Hansen and P. Salamon. Neural Network Ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, October 1990.

[83] M. N. Haque, M. N. Noman, R. Berretta, and P. Moscato. Optimising Weights for Heterogeneous Ensemble of Classifiers with Differential Evolution. In *Congress on Evolutionary Computation*, pages 233–240. IEEE, 2016.

[84] S. Hashem. Optimal Linear Combinations of Neural Networks. *Neural Networks*, 10(4): 599–614, June 1997.

[85] M. Hauschild and M. Pelikan. An Introduction and Survey of Estimation of Distribution Algorithms. *Swarm and Evolutionary Computation*, 1(3):111–128, September 2011.

[86] L. C. Hernández, A. M. Hernández, G. M. C. Cardoso, and Y. M. Jiménez. Genetic Algorithms with Diversity Measures to build Classifier Systems. *Investigación Operacional*, 36(3):206–225, September 2015.

[87] H. Ishibuchi and T. Yamamoto. Evolutionary multiobjective optimization for generating an ensemble of fuzzy rule-based classifiers. In *Conference on Genetic and Evolutionary Computation*, pages 197–197. ACM, 2003.

[88] K. Jackowski. Fixed-size Ensemble Classifier System Evolutionarily Adapted to a Recurring Context with an Unlimited Pool of Classifiers. *Pattern Analysis and Applications*, 17(4): 709–724, November 2014.

[89] K. Jackowski. Adaptive Splitting and Selection Algorithm for Regression. *New Generation Computing*, 33(4):425–448, October 2015.

[90] K. Jackowski, B. Krawczyk, and M. Woźniak. Improved Adaptive Splitting and Selection: The Hybrid Training Method of a Classifier based on a Feature Space Partitioning. *International Journal of Neural Systems*, 24(3):1430007, January 2014.

[91] S. Joardar, A. Chatterjee, S. Bandyopadhyay, and U. Maulik. Multi-size Patch based Collaborative Representation for Palm Dorsa Vein Pattern Recognition by Enhanced Ensemble Learning with Modified Interactive Artificial Bee Colony Algorithm. *Engineering Applications of Artificial Intelligence*, 60:151–163, April 2017.

[92] L. Kaiping, C. Binglian, D. Yan, and H. Ying. A Genetic Neural Network Ensemble Prediction Model based on Locally Linear Embedding for Typhoon Intensity. In *Conference on Industrial Electronics and Applications*, pages 137–142. IEEE, 2013.

[93] M. N. Kapp, R. Sabourin, and P. Maupin. Adaptive Incremental Learning with an Ensemble of Support Vector Machines. In *International Conference on Pattern Recognition*, pages 4048–4051. IEEE, 2010.

[94] M. N. Kapp, R. Sabourin, and P. Maupin. A Dynamic Optimization Approach for Adaptive Incremental Learning. *International Journal of Intelligent Systems*, 26(11):1101–1124, July 2011.

[95] S. Karakatič, M. Heričko, and V. Podgorelec. Weighting and Sampling data for Individual Classifiers and Bagging with Genetic Algorithms. In *International Joint Conference on Computational Intelligence*, pages 180–187. IEEE, 2015.

[96] A. Khamis, Y. Xu, Z. Y. Dong, and R. Zhang. Faster Detection of Microgrid Islanding Events using an Adaptive Ensemble Classifier. *IEEE Transactions on Smart Grid*, 9(3): 1889–1899, August 2016.

[97] K.-J. Kim and S.-B. Cho. DNA Gene Expression Classification with Ensemble Classifiers Optimized by Speciated Genetic Algorithm. *Pattern Recognition and Machine Intelligence*, 3776:649–653, December 2005.

[98] K.-J. Kim and S.-B. Cho. An Evolutionary Algorithm Approach to Optimal Ensemble Classifiers for DNA Microarray Data Analysis. *IEEE Transactions on Evolutionary Computation*, 12(3):377–388, June 2008.

[99] K.-J. Kim and S.-B. Cho. Evolutionary Ensemble of Diverse Artificial Neural Networks using Speciation. *Neurocomputing*, 71(7):1604–1618, March 2008.

[100] K.-J. Kim and S.-B. Cho. Meta-classifiers for High-dimensional, Small Sample Classification for Gene Expression Analysis. *Pattern Analysis and Applications*, 18(3):553–569, May 2015.

[101] Y. Kim, W. N. Street, and F. Menczer. Meta-evolutionary Ensembles. In *International Joint Conference on Neural Networks*, pages 2791–2796. IEEE, 2002.

[102] S. Kiranyaz, T. Ince, M. Zabihi, and D. Ince. Automated Patient-specific Classification of Long-term Electroencephalography. *Journal of Biomedical Informatics*, 49:16–31, June 2014.

[103] J. Kittler, M. Hatef, R. P. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, March 1998.

[104] A. H.-R. Ko, R. Sabourin, and A. d. S. Britto Jr. Evolving Ensemble of Classifiers in Random Subspace. In *Conference on Genetic and Evolutionary Computation*, pages 1473–1480. ACM, 2006.

[105] P. Kordík, J. Černý, and T. Frýda. Discovering Predictive Ensembles for Transfer Learning and Meta-learning. *Machine Learning*, 107:177–207, December 2018.

[106] S. B. Kotsiantis. Bagging and Boosting Variants for Handling Classifications Problems: A Survey. *The Knowledge Engineering Review*, 29(1):78–100, August 2014.

[107] B. Krawczyk and G. Schaefer. Breast Thermogram Analysis using Classifier Ensembles and Image Symmetry Features. *IEEE Systems Journal*, 8(3):921–928, October 2014.

[108] B. Krawczyk and M. Woźniak. Evolutionary Cost-sensitive Ensemble for Malware Detection. In *SOCO/CISIS/ICEUTE*, pages 433–442. Springer, 2014.

[109] B. Krawczyk, G. Schaefer, and M. Woźniak. A Cost-sensitive Ensemble Classifier for Breast Cancer Classification. In *International Symposium on Applied Computational Intelligence and Informatics*, pages 427–430. IEEE, 2013.

[110] B. Krawczyk, M. Woźniak, and G. Schaefer. Cost-sensitive Decision Tree Ensembles for Effective Imbalanced Classification. *Applied Soft Computing*, 14:554–562, January 2014.

[111] B. Krawczyk, G. Schaefer, and M. Woźniak. A hybrid Cost-sensitive Ensemble for Imbalanced Breast Thermogram Classification. *Artificial Intelligence in Medicine*, 65(3): 219–227, November 2015.

[112] B. Krawczyk, M. Galar, Ł. Jeleń, and F. Herrera. Evolutionary Undersampling Boosting for Imbalanced Classification of Breast Cancer Malignancy. *Applied Soft Computing*, 38: 714–726, January 2016.

[113] B. Krawczyk et al. Ensemble Learning for Data Stream Analysis: A Survey. *Information Fusion*, 37:132–156, September 2017.

[114] M. Krithikaa and R. Mallipeddi. Differential Evolution with an Ensemble of Low-quality Surrogates for Expensive Optimization Problems. In *Congress on Evolutionary Computation*, pages 78–85. IEEE, 2016.

[115] A. Krogh and J. Vedelsby. Neural Network Ensembles, Cross Validation, and Active Learning. *Advances in Neural Information Processing Systems*, 7:231–238, January 1995.

[116] G. Kumar and K. Kumar. Design of an Evolutionary Approach for Intrusion Detection. *The Scientific World Journal*, 2013:1–14, November 2013.

[117] M. Kumar, M. Husian, N. Upreti, and D. Gupta. Genetic algorithm: Review and application. *International Journal of Information Technology*, 2(2):451–454, 2010.

[118] L. I. Kuncheva and C. J. Whitaker. Measures of Diversity in Classifier Ensembles and their Relationship with the Ensemble Accuracy. *Machine learning*, 51(2):181–207, May 2003.

[119] S. E. Lacy, M. A. Lones, and S. L. Smith. A Comparison of Evolved Linear and Non-linear Ensemble Vote Aggregators. In *Congress on Evolutionary Computation*, pages 758–763. IEEE, 2015.

[120] S. E. Lacy, M. A. Lones, and S. L. Smith. Forming Classifier Ensembles with Multimodal Evolutionary Algorithms. In *Congress on Evolutionary Computation*, pages 723–729. IEEE, 2015.

[121] J.-C. Lévesque, A. Durand, C. Gagné, and R. Sabourin. Multi-objective Evolutionary Optimization for Generating Ensembles of Classifiers in the ROC Space. In *Conference on Genetic and Evolutionary Computation*, pages 879–886. ACM, 2012.

[122] W. S. Liew, C. K. Loo, and T. Obo. Optimizing FELM ensembles using GA-BIC. In *Joint World Congress of International Fuzzy Systems Association and International Conference on Soft Computing and Intelligent Systems*, pages 1–6. IEEE, 2017.

[123] T. P. F. Lima and T. B. Ludermir. Differential Evolution and Meta-learning for Dynamic Ensemble of Neural Network Classifiers. In *International Joint Conference on Neural Networks*, pages 1–5. IEEE, 2015.

[124] K. Liu, M. Tong, S. Xie, and Z. Zeng. Fusing Decision Trees based on Genetic Programming for Classification of Microarray Datasets. In *International Conference on Intelligent Computing*, pages 126–134. Springer, 2014.

[125] K.-H. Liu, D.-S. Huang, and J. Zhang. Microarray Data Prediction by Evolutionary Classifier Ensemble System. In *Congress on Evolutionary Computation*, pages 634–637. IEEE, 2007.

[126] K.-H. Liu, B. Li, J. Zhang, and J.-X. Du. Ensemble Component Selection for Improving ICA based Microarray Data Prediction Models. *Pattern Recognition*, 42(7):1274–1283, July 2009.

[127] K.-H. Liu, M. Tong, S.-T. Xie, and V. T. Yee Ng. Genetic Programming based Ensemble System for Microarray Data Classification. *Computational and Mathematical Methods in Medicine*, 2015:1–11, February 2015.

[128] N. Liu et al. Evolutionary Voting-based Extreme Learning Machines. *Mathematical Problems in Engineering*, 2014:1–7, August 2014.

[129] Y. Liu et al. Ensemble of Surrogates with an Evolutionary Multi-agent System. In *International Conference on Computer Supported Cooperative Work in Design*, pages 521–525. IEEE, 2017.

[130] M. A. Lones et al. Evolving Classifiers to Recognize the Movement Characteristics of Parkinson's Disease Patients. *IEEE Transactions on Evolutionary Computation*, 18(4): 559–576, August 2014.

[131] N. Ma, H. Fujita, Y. Zhai, and S. Wang. Ensembles of Fuzzy Cognitive Map Classifiers Based on Quantum Computation. *Acta Polytechnica Hungarica*, 12(4):7–26, 2015.

[132] S. Mabu, M. Obayashi, and T. Kuremoto. Ensemble Learning of Rule-based Evolutionary Algorithm using Multi Layer Perceptron for Stock Trading Models. In *Joint International Conference on Soft Computing and Intelligent Systems and International Symposium on Advanced Intelligent Systems*, pages 624–629. IEEE, 2014.

[133] S. Mabu, M. Obayashi, and T. Kuremoto. Ensemble Learning of Rule-based Evolutionary Algorithm using Multi-layer Perceptron for Supporting Decisions in Stock Trading Problems. *Applied Soft Computing*, 36:357–367, November 2015.

[134] G. Mauša and T. G. Grbac. Co-evolutionary Multi-population Genetic Programming for Classification in Software Defect Prediction: An Empirical Case Study. *Applied Soft Computing*, 55:331–351, June 2017.

[135] N. Mehdiyev, J. Krumeich, D. Werth, and P. Loos. Sensor Event Mining with Hybrid Ensemble Learning and Evolutionary Feature Subset Selection Model. In *International Conference on Big Data*, pages 2159–2168. IEEE, 2015.

[136] J. a. Mendes-Moreira, C. Soares, A. M. Jorge, and J. F. de Sousa. Ensemble Approaches for Regression: A Survey. *ACM Computing Surveys*, 45(1):10:1–10:40, November 2012.

[137] M. Milliken, Y. Bi, L. Galway, and G. Hawe. Multi-objective Optimization of Base Classifiers in StackingC by NSGA-II for Intrusion Detection. In *Symposium Series on Computational Intelligence*, pages 1–8. IEEE, 2016.

[138] S. C. Neoh et al. Intelligent Facial Emotion Recognition using a Layered Encoding Cascade Optimization Model. *Applied Soft Computing*, 34:72–93, September 2015.

[139] T. Obo, N. Kubota, and C. K. Loo. Evolutionary Ensemble Learning of Fuzzy Randomized Neural Network for Posture Recognition. In *World Automation Congress*, pages 1–6. IEEE, 2016.

[140] S. Oehmcke, J. Heinermann, and O. Kramer. Analysis of Diversity Methods for Evolutionary Multi-objective Ensemble Classifiers. In *European Conference on the Applications of Evolutionary Computation*, pages 567–578. Springer, 2015.

[141] V. K. Ojha, K. Jackowski, A. Abraham, and V. Snášel. Feature Selection and Ensemble of Regression Models for Predicting the Protein Macromolecule Dissolution Profile. In *World Congress on Nature and Biologically Inspired Computing*, pages 121–126. IEEE, 2014.

[142] V. K. Ojha, K. Jackowski, A. Abraham, and V. Snášel. Dimensionality Reduction, and Function Approximation of Poly (Lactic-co-glycolic acid) Micro-and Nanoparticle Dissolution Rate. *International Journal of Nanomedicine*, 10:1119, February 2015.

[143] V. K. Ojha, A. Abraham, and V. Snášel. Ensemble of Heterogeneous Flexible Neural Trees using Multiobjective Genetic Programming. *Applied Soft Computing*, 52:909–924, March 2017.

[144] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore. Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science. In *Conference on Genetic and Evolutionary Computation*, pages 485–492. ACM, 2016.

[145] J. A. Olvera-López, J. A. Carrasco-Ochoa, J. Martínez-Trinidad, and J. Kittler. A Review of Instance Selection Methods. *Artificial Intelligence Review*, 34(2):133–143, May 2010.

[146] A. Onan, S. Korukoğlu, and H. Bulut. A Multiobjective Weighted Voting Ensemble Classifier based on Differential Evolution Algorithm for Text Sentiment Classification. *Expert Systems with Applications*, 62:1–16, November 2016.

[147] D. Opitz and R. Maclin. Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research*, 11:169–198, August 1999.

[148] D. W. Opitz. Feature Selection for Ensembles. In *National Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence Conference*, page 384. American Association for Artificial Intelligence, 1999.

[149] N. C. Oza and K. Tumer. Classifier Ensembles: Select Real-world Applications. *Information Fusion*, 9(1):4–20, January 2008.

[150] C. Pagano, E. Granger, R. Sabourin, and D. O. Gorodnichy. Detector Ensembles for Face Recognition in Video Surveillance. In *International Joint Conference on Neural Networks*, pages 1–8. IEEE, 2012.

[151] E. Parhizkar and M. Abadi. BeeOWA: A Novel Approach based on ABC Algorithm and Induced OWA operators for Constructing One-class Classifier Ensembles. *Neurocomputing*, 166:367–381, October 2015.

[152] E. Parhizkar and M. Abadi. OC-WAD: A One-class Classifier Ensemble Approach for Anomaly Detection in Web Traffic. In *Iranian Conference on Electrical Engineering*, pages 631–636. IEEE, 2015.

[153] C. Park and S.-B. Cho. Evolutionary Ensemble Classifier for Lymphoma and Colon Cancer Classification. In *Congress on Evolutionary Computation*, pages 2378–2385. IEEE, 2003.

[154] C. Park and S.-B. Cho. Evolutionary Computation for Optimal Ensemble Classifier in Lymphoma Cancer Classification. *Foundations of Intelligent Systems*, 2871:521–530, October 2003.

[155] F. Pedregosa et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, October 2011.

[156] A. Peimankar, S. J. Weddell, T. Jalal, and A. C. Lapthorn. Ensemble Classifier Selection using Multi-objective PSO for Fault Diagnosis of Power Transformers. In *Congress on Evolutionary Computation*, pages 3622–3629. IEEE, 2016.

[157] A. Peimankar, S. J. Weddell, T. Jalal, and A. C. Lapthorn. Evolutionary Multi-Objective Fault Diagnosis of Power Transformers. *Swarm and Evolutionary Computation*, 36:62–75, October 2017.

[158] Z. K. Pourtaheri and S. H. Zahiri. Ensemble Classifiers with Improved Overfitting. In *Conference on Swarm Intelligence and Evolutionary Computation*, pages 93–97. IEEE, 2016.

[159] A. Rahman and B. Verma. Cluster Based Ensemble Classifier Generation by Joint Optimization of Accuracy and Diversity. *International Journal of Computational Intelligence and Applications*, 12(4):1340003, December 2013.

[160] A. Rahman and B. Verma. Cluster Oriented Ensemble Classifiers using Multi-objective Evolutionary Algorithm. In *International Joint Conference on Neural Networks*, pages 1–6. IEEE, 2013.

[161] A. Rahman and B. Verma. Ensemble Classifier Generation using Non-uniform Layered Clustering and Genetic Algorithm. *Knowledge-Based Systems*, 43:30–42, May 2013.

[162] T. Rapakoulia et al. EnsembleGASVR: a Novel Ensemble Method for Classifying Missense Single Nucleotide Polymorphisms. *Bioinformatics*, 30(16):2324–2333, August 2014.

[163] M. D. Redel-Macías et al. Ensembles of Evolutionary Product Unit or RBF Neural Networks for the Identification of Sound for Pass-by Noise Test in Vehicles. *Neurocomputing*, 109: 56–65, June 2013.

[164] P. J. Roebber. Adaptive Evolutionary Programming. *Monthly Weather Review*, 143(5): 1497–1505, May 2015.

[165] L. Rokach. Ensemble-based Classifiers. *Artificial Intelligence Review*, 33(1):1–39, November 2010.

[166] A. Rosales-Pérez et al. Multi-objective Model Type Selection. *Neurocomputing*, 146:83–94, December 2014.

[167] A. Rosales-Pérez et al. An Evolutionary Multi-Objective Model and Instance Selection for Support Vector Machines with Pareto-based Ensembles. *IEEE Transactions on Evolutionary Computation*, 21(6):863–877, March 2017.

[168] O. Sagi and L. Rokach. Ensemble Learning: A Survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4), January 2018.

[169] S. Saha, S. Mitra, and R. K. Yadav. A Multiobjective based Automatic Framework for Classifying Cancer-microRNA Biomarkers. *Gene Reports*, 4:91–103, September 2016.

[170] R. Saleh, H. Farsi, and S. H. Zahiri. Ensemble Classification of PolSAR Data using Multi-objective Heuristic Combination Rule. In *Conference on Swarm Intelligence and Evolutionary Computation*, pages 88–92. IEEE, 2016.

[171] S. K. K. Santu, M. M. Rahman, M. M. Islam, and K. Murase. Towards better Generalization in Pittsburgh Learning Classifier Systems. In *Congress on Evolutionary Computation*, pages 1666–1673. IEEE, 2014.

[172] G. Schaefer. Evolutionary Optimisation of Classifiers and Classifier Ensembles for Cost-sensitive Pattern Recognition. In *International Symposium on Applied Computational Intelligence and Informatics*, pages 343–346. IEEE, 2013.

[173] R. E. Schapire. A Brief Introduction to Boosting. In *International Joint Conference on Artificial Intelligence*, pages 1401–1406. European Association for Artificial Intelligence, 1999.

[174] C. D. Schuman, J. D. Birdwell, and M. E. Dean. Spatiotemporal Classification using Neuroscience-inspired Dynamic Architectures. *Procedia Computer Science*, 41:89–97, November 2014.

[175] P. Shunmugapriya and S. Kanmani. Optimization of Stacking Ensemble Configurations through Artificial Bee Colony Algorithm. *Swarm and Evolutionary Computation*, 12:24–32, October 2013.

[176] U. K. Sikdar, A. Ekbal, and S. Saha. Differential Evolution based Feature Selection and Classifier Ensemble for Named Entity Recognition. In *International Conference on Computational Linguistics*, pages 2475–2490. International Committee on Computational Linguistics, 2012.

[177] U. K. Sikdar, A. Ekbal, and S. Saha. Differential Evolution based Mention Detection for Anaphora Resolution. In *India Conference*, pages 1–6. IEEE, 2013.

[178] U. K. Sikdar, A. Ekbal, and S. Saha. Differential Evolution based Multiobjective Optimization for Biomedical Entity Extraction. In *International Conference on Advances in Computing, Communications and Informatics*, pages 1039–1044. IEEE, 2014.

[179] U. K. Sikdar, A. Ekbal, and S. Saha. Entity Extraction in Biochemical Text using Multiobjective Optimization. *Computación y Sistemas*, 18(3):591–602, February 2014.

[180] U. K. Sikdar, A. Ekbal, and S. Saha. MODE: Multiobjective Differential Evolution for Feature Selection and Classifier Ensemble. *Soft Computing*, 19(12):3529–3549, January 2015.

[181] U. K. Sikdar, A. Ekbal, and S. Saha. A Generalized Framework for Anaphora Resolution in Indian Languages. *Knowledge-Based Systems*, 109:147–159, October 2016.

[182] I. Singh, K. Sanwal, and S. Praveen. Breast Cancer Detection using two-fold Genetic Evolution of Neural Network Ensembles. In *International Conference on Data Science and Engineering*, pages 1–6. IEEE, 2016.

[183] C. D. Stefano, F. Fontanella, G. Folino, and A. Freca. A Bayesian approach for Combining Ensembles of GP Classifiers. In *International Workshop on Multiple Classifier Systems*, pages 26–35. Springer, 2011.

[184] N. Tabassum and T. Ahmed. A Theoretical Study on Classifier Ensemble Methods and its Applications. In *International Conference on Computing for Sustainable Global Development*, pages 374–378. IEEE, 2016.

[185] C. J. Tan, C. P. Lim, and Y.-N. Cheah. A Multi-objective Evolutionary Algorithm-based Ensemble Optimizer for Feature Selection and Classification with Neural Network Models. *Neurocomputing*, 125:217–228, February 2014.

[186] H. L. Tang et al. The Reading of Components of Diabetic Retinopathy: An Evolutionary approach for Filtering normal Digital Fundus Imaging in Screening and Population based Studies. *PloS One*, 8(7):e66730, July 2013.

[187] J. Tian and N. Feng. Adaptive Generalized Ensemble Construction with Feature Selection and its Application in Recommendation. *International Journal of Computational Intelligence Systems*, 7(sup2):35–43, July 2014.

[188] K. Trawiński, O. Cordón, A. Quirin, and L. Sánchez. Multiobjective Genetic Classifier Selection for Random Oracles Fuzzy Rule-based Classifier Ensembles: How Beneficial is the Additional Diversity? *Knowledge-Based Systems*, 54:3–21, December 2013.

[189] K. Trawiński, O. Cordón, and A. Quirin. Embedding Evolutionary Multiobjective Optimization into Fuzzy Linguistic Combination method for Fuzzy Rule-based Classifier Ensembles. In *International Conference on Fuzzy Systems*, pages 1968–1975. IEEE, 2014.

[190] S. K. Trivedi and S. Dey. A Study of Ensemble based Evolutionary Classifiers for Detecting Unsolicited Emails. In *Conference on Research in Adaptive and Convergent Systems*, pages 46–51. ACM, 2014.

[191] A. Tsakonas. An analysis of Accuracy-diversity Trade-off for Hybrid Combined System with Multiobjective Predictor Selection. *Applied Intelligence*, 40(4):710–723, January 2014.

[192] A. Tsakonas and B. Gabrys. A Fuzzy Evolutionary Framework for Combining Ensembles. *Applied Soft Computing*, 13(4):1800–1812, April 2013.

[193] E. Vaiciukynas et al. Fusion of Voice Signal Information for Detection of Mild Laryngeal Pathology. *Applied Soft Computing*, 18:91–103, May 2014.

[194] K. Veeramachaneni, O. Derby, D. Sherry, and U.-M. O'Reilly. Learning Regression Ensembles with Genetic Programming at Scale. In *Conference on Genetic and Evolutionary Computation*, pages 1117–1124. ACM, 2013.

[195] S. Vega-Pons and J. Ruiz-Shulcloper. A survey of Clustering Ensemble Algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, 25(3):337–372, May 2011.

[196] S. Vluymans, I. Triguero, C. Cornelis, and Y. Saeys. EPRENNID: An Evolutionary Prototype Reduction based Ensemble for Nearest Neighbor Classification of Imbalanced Data. *Neurocomputing*, 216:596–610, December 2016.

[197] B. Vukobratović and R. Struharik. Hardware Acceleration of Nonincremental Algorithms for the Induction of Decision Trees. In *Telecommunication Forum*, pages 1–8. IEEE, 2017.

[198] D. Wang and M. Alhamdoosh. Evolutionary Extreme Learning Machine Ensembles with Size Control. *Neurocomputing*, 102:98–110, February 2013.

[199] Y.-W. Wen and C.-K. Ting. Learning Ensemble of Decision Trees through Multifactorial Genetic Programming. In *Congress on Evolutionary Computation*, pages 5293–5300. IEEE, 2016.

[200] S. Winkler et al. Data-based Prediction of Sentiments using Heterogeneous Model Ensembles. *Soft Computing*, 19(12):3401–3412, July 2015.

[201] M. Wistuba, N. Schilling, and L. Schmidt-Thieme. Automatic Frankensteining: Creating Complex Ensembles Autonomously. In *International Conference on Data Mining*, pages 741–749. SIAM, 2017.

[202] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2016.

[203] D. H. Wolpert. Stacked Generalization. *Neural Networks*, 5(2):241–259, July 1992.

[204] D. H. Wolpert and W. G. Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[205] W. L. Woon and O. Kramer. Enhanced SVR Ensembles for Wind Power Prediction. In *International Joint Conference on Neural Networks*, pages 2743–2748. IEEE, 2016.

[206] M. Wozniak. Evolutionary Approach to Produce Classifier Ensemble based on Weighted Voting. In *World Congress on Nature and Biologically Inspired Computing*, pages 648–653. IEEE, 2009.

[207] J. a. C. Xavier-Júnior, A. A. Freitas, A. Feitosa-Neto, and T. B. Ludermir. A novel Evolutionary Algorithm for Automated Machine Learning Focusing on Classifier Ensembles. In *Brazilian Conference on Intelligent Systems*, São Paulo, Brazil, 2018. IEEE.

[208] H. Xu, C. Caramanis, and S. Mannor. Sparse Algorithms are not Stable: A no-free-lunch Theorem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1):187–193, January 2012.

[209] R. R. Yager. On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decisionmaking. *IEEE Transactions on systems, Man, and Cybernetics*, 18(1):183–190, 1988.

[210] X. Yao and M. M. Islam. Evolving Artificial Neural Network Ensembles. *IEEE Computational Intelligence Magazine*, 3(1):31–42, February 2008.

[211] A. Zagorecki. Feature Selection for Naive Bayesian Network Ensemble using Evolutionary Algorithms. In *Federated Conference on Computer Science and Information Systems*, pages 381–385. IEEE, 2014.

[212] L. Zhang, K. Mistry, S. C. Neoh, and C. P. Lim. Intelligent Facial Emotion Recognition using Moth-firefly Optimization. *Knowledge-Based Systems*, 111:248–267, November 2016.

[213] W. Zhang et al. A Combined Model based on CEEMDAN and Modified Flower Pollination Algorithm for Wind Speed Forecasting. *Energy Conversion and Management*, 136:439–451, March 2017.

[214] Y. Zhang, H. Zhang, J. Cai, and B. Yang. A Weighted Voting Classifier based on Differential Evolution. *Abstract and Applied Analysis*, 2014(1):1–6, May 2014.

[215] Y. Zhang, B. Liu, and F. Yang. Differential Evolution Based Selective Ensemble of Extreme Learning Machine. In *Trustcom/BigDataSE/ISPA*, pages 1327–1333. IEEE, 2016.

[216] Y. Zhang, B. Liu, J. Cai, and S. Zhang. Ensemble Weighted Extreme Learning Machine for Imbalanced Data Classification based on Differential Evolution. *Neural Computing and Applications*, 28(1):259–267, May 2017.