Knowledge-Based Systems 254 (2022) 109669

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Minimally overfitted learners: A general framework for ensemble learning

Víctor Aceña [a,b,*], Isaac Martín de Diego [a], Rubén R. Fernández [a], Javier M. Moguerza [a]

[a] Rey Juan Carlos University, Data Science Laboratory, c/ Tulipán, s/n, 28933, Móstoles, Spain[1]
[b] MADOX VIAJES, Calle de Cantabria, 10, 28939, Arroyomolinos, Spain[2]

ARTICLE INFO

Article history:
Received 19 April 2022
Received in revised form 20 July 2022
Accepted 7 August 2022
Available online 12 August 2022

Keywords:
Ensemble
Generative ensembles
Re-sampling
Bagging
Random forest

ABSTRACT

The combination of *Machine Learning* (*ML*) algorithms is a solution for constructing stronger predictors than a single one. However, some approximations suggest that combining unstable algorithms provides better results than combining stable algorithms. For instance, Generative ensembles, based on re-sampling techniques, have demonstrated high performance by fusing the information of unstable base learners. Random Forest and Gradient Boosting are two well-known examples, both combining Decision Trees and providing better predictions than those obtained using a single tree. However, such successful results have not been achieved by assembling stable algorithms. This paper introduces the notion of limited learner and a new ensemble general framework called *Minimally Overfitted Ensemble* (*MOE*), a re-sampling-based ensemble approach that constructs slightly overfitted-based learners. The proposed framework works well with stable and unstable base algorithms, thanks to a *Weighted RAndom Bootstrap* (*WRAB*) sampling that provides the necessary diversity for the stable base algorithms. A hyperparameter analysis of the proposal is carried out on artificial data. Besides, its performance is evaluated on real datasets against well-known *ML* methods. The results confirm that the *MOE* framework works successfully using stable and unstable base algorithms, improving in most cases the predictive ability of single *ML* models and other ensemble methods.

1. Introduction

Machine Learning (*ML*) is a field of Artificial Intelligence comprising algorithms which provide predictions by learning from (past) data. These algorithms implement models that are used to make inferences about new data. An important goal in these models is to make the error in the new data as low as possible. This error is known as the generalization error and it can be broken down into two types of errors: reducible and irreducible. The reducible error is avoidable, whereas the irreducible error will always be present. Therefore, the generalization error can be reduced by decreasing the reducible error.

The reducible error can also be split into two components, the *Bias* and *Variance* errors. The *Bias* is the difference between the prediction of the model and the actual value. The *Variance* is the variability in the predictions when the model is trained with different training sets. There is a trade-off between the *Bias* and *Variance* errors; when one of them increases, the other decreases and vice versa. This fact becomes more evident when the high *Bias* corresponds to underfitting, and the high *Variance* implies overfitting. However, the *Bias-Variance* decomposition for classification problems is not straightforward to assess, especially for ensemble methods [1].

Whenever a *ML* model is adjusted, the aim is to achieve a low generalization error. Thus, the model fits in some point between overfitting and underfitting, often referred to as well-fitted. There are many levels of overfitting that range from strongly overfitted to well-fitted, and the same applies to underfitting. Both scenarios lead to undesirable models. Fig. 1 illustrates the concept of different fitting levels for the well-known half-moons dataset.

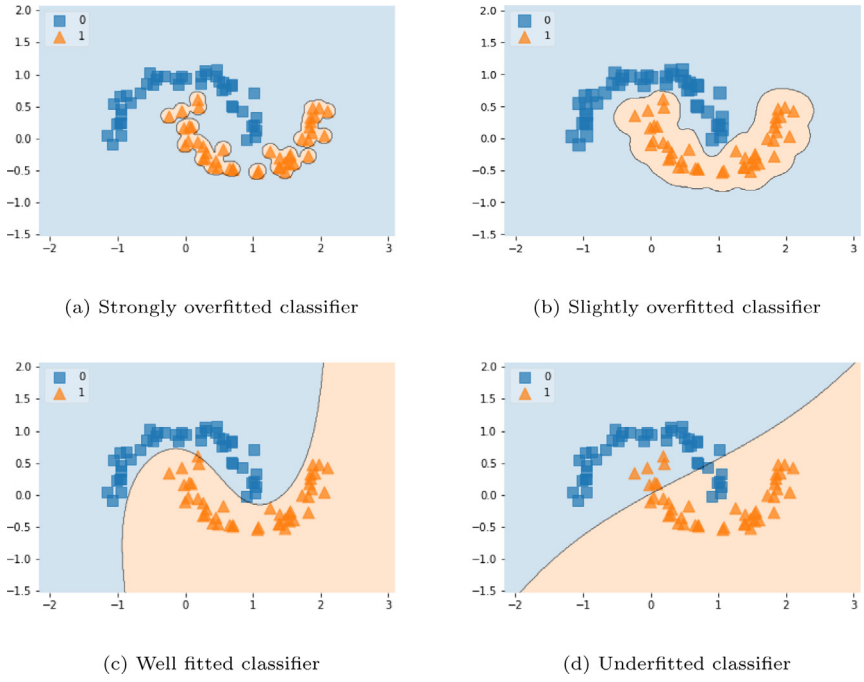
Ideally, a well-fitted *ML* model gets the same error in the train and test sets, and additionally, it matches its irreducible error. In practice, it is desirable that the training and test errors are both small and similar. On the one hand, when the training error is much lower than the test error the overfitting phenomenon appears. On the other hand, underfitting is reached when both, the train and test errors, are high. A desirable *ML* model seeks for a balance between these two scenarios.

Ensembles are a group of *ML* techniques that combine several reduced models (referred to as single learners) rather than an

* Corresponding author at: Rey Juan Carlos University, Data Science Laboratory, c/ Tulipán, s/n, 28933, Móstoles, Spain.
E-mail addresses: victor.acena@urjc.es (V. Aceña), isaac.martin@urjc.es (I. Martín de Diego), ruben.rodriguez@urjc.es (R. R. Fernández), javier.moguerza@urjc.es (J. M. Moguerza).

[1] www.datasciencelab.es
[2] www.madoxviajes.com

https://doi.org/10.1016/j.knosys.2022.109669
0950-7051/



(a) Strongly overfitted classifier (b) Slightly overfitted classifier

(c) Well fitted classifier (d) Underfitted classifier

Fig. 1. Representation of different overfitting level through diverse *Support Vector Machine SVM* hyperparameter settings. There are four decision functions that represent strongly overfitted, slightly overfitted, well-fitted, and underfitted classifiers, respectively.

overall single model. The best known ensemble methods are *Bootstrap aggregating* (*Bagging*) [2] and *Boosting* [3]. The performance of ensemble-based methods is generally better than single (and probably more complex) models. The key reason of this performance improvement in *Bagging* and *Boosting* ensembles is the diversity of the single learners. This diversity is obtained by perturbing the learning set. In *Bagging*, the diversity is achieved by training each single learner on a dataset made of n bootstrap samples and then combining the predictions using a voting scheme. In *Boosting*-based ensembles, the diversity is achieved by re-weighting the learning set. These weights are calculated based on the errors of each single learner. In *Boosting*, these single learners are referred to as weak learners. Intuitively, a weak learner is a *ML* model that provides predictions just slightly better than random [4]. Finally, regarding the *Bias* and the *Variance*, *Bagging* is designed to reduce the *Variance*, whereas *Boosting* reduces the *Bias* in the first iterations, and the *Variance* in the last ones.

The most widespread instances of *Bagging* and *Boosting* are *Random Forest* (*RF*) [5] and *eXtreme Gradient Boosting* (*XGBoost*) [6], respectively. Both use a *Decision Tree* (*DT*) as base learner. The main quality that makes *DT*s a really good base learner is their instability. A *ML* model is considered unstable if a small change in the input (learning set) produces major changes in the output (prediction). It is well known that unstable models such as *DT*s or *Neural Networks* work well in ensembles [7,8], contrary to stable models like *k-Nearest Neighbor* (*kNN*), *SVM* or *Linear Discriminant Analysis* (*LDA*).

An unstable model has high *Variance* and low *Bias*. In other words, unstable models tend to overfit. Regarding overfitting, it has been empirically seen that no-pruned trees perform better than pruned trees for *Bagging* [9]. Particularly, the *RF* algorithm [5] grows each tree as large as possible with no pruning. More recently, [10] verifies that bigger trees perform better in *RF*. Thus, a certain level of overfitting is beneficial for *Bagging*-based ensembles. This fact will be the primary motivation for the present proposal. The success of this work lies in how the overfitted learners perform in Bagging-based ensembles and in the production of appropriate samples for training any base algorithm that can produce satisfactory results.

In this paper, we present the concept of limited learner and a novel generalized ensemble learning framework called *Minimally Overfitted Ensemble* (*MOE*). The limited learners are base learners that overfit the training sample. The proposed approach is based on the minimum overfitting of single learners, a particular case of limited learners. Furthermore, we propose the *Weighted RAndom Bootstrap* (*WRAB*) sampling, a new re-sampling method that provides the necessary diversity to construct an ensemble method with any *ML* algorithm (stable or unstable). The basic predictors obtained from this construction are limited learners. The proposal is evaluated in eighteen real datasets and its performance is compared with state-of-the-art *ML* models, including *RF* as the top representative *Bagging* method.

The remainder of the paper is outlined as follows. In Section 2 the different ensemble methods and their taxonomy are presented. Section 3 introduces the limited learner concept. The new ensemble framework is detailed in Section 4. The experiments to validate the proposal and a brief discussion with the lessons learned are in Section 5. Finally, Section 6 concludes and sketches future research guidelines.

2. Generative ensembles

A large variety of combination schemes and ensemble techniques are available in the *ML* literature. These techniques can be grouped in different ways depending on the selected criterion. For the sake of simplicity, this work follows the ensemble taxonomy proposed in [11,12]. In this taxonomy, there are two fundamental levels: Non-generative and Generative ensemble methods. The distinction between these two levels depends on the prevalence of generation and combination methods.

Non-generative ensembles focus on the combination of the base learners. Thus, each learner is previously fitted to achieve a high performance. Some examples of these type of ensembles are methods that produce a prediction by majority voting [13, 14], methods that combine the probabilistic result by a Bayesian

Table 1
The most popular Generative ensembles *Bagging* and *Boosting* based.

Re-sampling method	Algorithm	Key features
Bootstrap aggregating (*Bagging*)	*Random Forest* (*RF*)	Subset of random features
	Extra Trees (*ET*)	Subset of random features, randomly cut-points
Boosting	*Adaptive Boosting* (*AdaBoost*)	Weighted prediction
	Gradient Boosting (*GB*)	Weighted prediction, steepest descent minimization loss function
	LightGBM	Gradient-based One-Side Sampling, Exclusive Feature Building
	eXtreme Gradient Boosting (*XGBoost*)	Regularization, weighted quantile sketch

decision rule [15,16], and methods that select the base learners best subset [17]. Notice that, in all these examples, the emphasis is placed on how the learners are merged [18]. Non-generative ensembles have demonstrated that combining information from the appropriate learner [19] outperforms the results obtained from other *ML* single methods like *SVM*s [20] or *DT*s [21] in specific problems like cross-domain collaborative filtering.

Generative ensembles generate a pool of learners by perturbing the base algorithm or the learning set for each base learner, focusing on the diversity and performance of the learners. Thus, the emphasis is on how the base learners are built, not the combination of the individual results. Instances of Generative ensembles are re-sampling methods [2,3], feature selection [22], mixture of experts [23], and highly randomized methods [5,24]. All these methods are designed to ensure the diversity of the base learners. Furthermore, a necessary and sufficient condition for Generative ensembles to perform better than a single learner is that the individual learners are accurate (perform better than random) and diverse [25].

The most popular Generative ensemble are *Bagging* and *Boosting*. They have been widely studied and employed over the years for classification or regression tasks [26,27]. There are several well-known instances of *Boosting*-based ensembles such as *AdaBoost* [28], *LightGBM* [29], and *XGBoost* [6], whereas in *Bagging*-based ensembles the default choice is *RF* [5]. All these method have something in common, they are re-sampling based and use a *Decision Tree* (*DT*) as base learner. The main advantages of *DT*s are their robustness against outliers and noise, and their ability to capture non-linear patterns. On the other hand, *DT*s are unstable and they usually overfit. However, these downsides alongside their advantages make *DT*s a good choice for base learners in Generative ensembles [30].

The hyperparameters in *Boosting*-based methods have a very high influence on the performance of the method. In contrast, the default *RF* configuration, in the most common implementations, achieves good accuracy and can be improved by just slightly modifying the number of estimators or the number of features included in each base learner. Because of the advantages of *RF*, there are *Bagging* attempts using stable-based learners to emulate its performance. Table 1 shows a summary of the most common Generative ensembles and their main characteristics. For example, in [31], *Support Vector Machine* (*SVM*) ensembles are considered. In such a case, linear *Bagging SVM* outperforms *Gaussian Bagging SVM*, as expected since linear kernels produce a *SVM* more unstable than Gaussian kernels. Other approaches found improvements by modifying the training data using different kinds of under-sampling and over-sampling [32,33]. These types of sampling can be applied to ensembles using as base learners *Bayesian Neural Networks* [34], where the traditional diversity and re-balancing techniques work together to improve the performance of a single *Bayesian Neural Network*.

There are interesting alternatives to *Boosting* and *Bagging* in the literature, such as *Weight Aggregation* (*Wagging*) [9], which perturbs the dataset by adding Gaussian noise to the vector of weights and thus modifying each replica. However, the performance of ensemble methods based on *Wagging* and *Bagging* is similar [9]. In addition, it increases the complexity of the model since one must consider the distribution parameters for adding the noise. More recent studies to assemble stable models have turned to new sampling *Bagging*-based methods to achieve the required diversity and accuracy in the different learners [35]. Nevertheless, the latter method is not generally applicable since it uses specific features of the base algorithm. Therefore, it is necessary to develop techniques to increase diversity to achieve effective ensembles, even for complicated to assemble base learners such as *k-Nearest Neighbor* (*kNN*) or *SVM*.

3. Limited learners

In this section, we introduce the concept of the limited learner. It is a learner that overfits the sample from which it has been constructed and predicts well enough.

Let us consider a learning set $\mathcal{L} = \{(\boldsymbol{x}_i, y_i) : i = 1, 2, \ldots, n\}$ where $\boldsymbol{x_i} \in \mathbb{R}^r$ ($r \in \mathbb{N}$) and $y_i \in \{-1, 1\}$. Suppose a sampling method (S), with replacement or not. Let be $\mathcal{L}_k^{(S)} = \{(\boldsymbol{x}'_i, y'_i) : i = 1, 2, \ldots, m\}$ the kth sample of the learning set, where $m \leq n$ and $\boldsymbol{x}'_i \in \mathbb{R}^{\leq r}$. Note that, if the sampling is over the features, there exists a projection $\phi : \mathbb{R}^r \longrightarrow \mathbb{R}^{\leq r}$ such that $\phi(\boldsymbol{x}_i) \longmapsto \boldsymbol{x}'_i$, $\forall(\boldsymbol{x}'_i, y'_i) \in \mathcal{L}_k^{(S)}$ where $(\boldsymbol{x}_i, y_i) \in \mathcal{L}$.

Thus, any learner is built using the sample $\mathcal{L}_k^{(S)}$ and it is denoted by $f_k : \mathbb{R}^{\leq r} \longrightarrow \{-1, 1\}$, a surjective function such that $f_k(\boldsymbol{x}'_i) \longmapsto y_i$ where $(\boldsymbol{x}'_i, y_i) \in \mathcal{L}_k^{(S)}$. In addition, the learner is tested in $\mathcal{T} \subseteq \mathcal{L} - \mathcal{L}_k^{(S)}$ (i.e., the *out-of-bag* (*OOB*) observations for the kth sample).

Definition 1 (*Random Predictor*)**.** The random predictor is a model that predicts according exclusively to the target probability distribution.

Definition 2 (*Dummy Learner*)**.** A dummy learner is a learner that returns the observed label for the instances in the training sample and it is a random predictor in new samples.

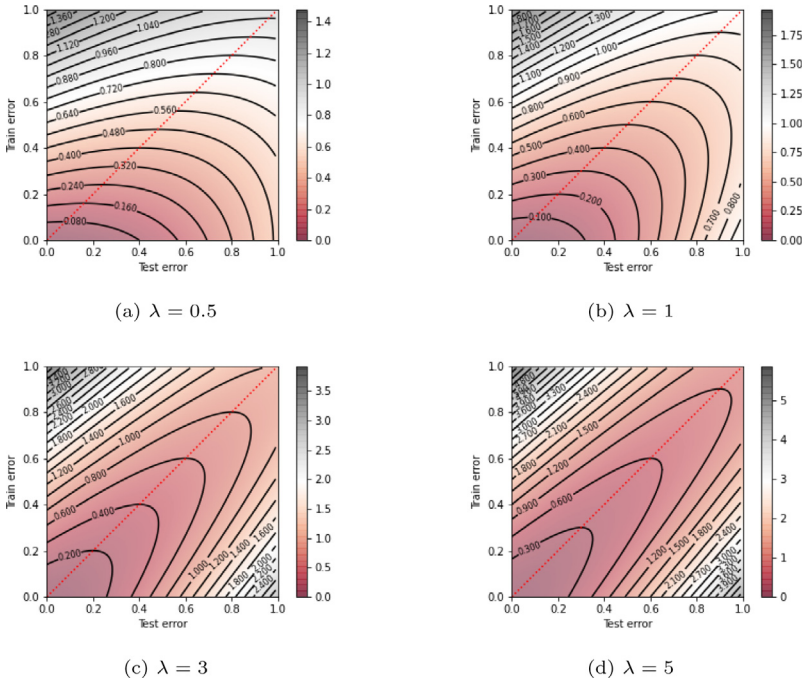
Definition 3 (*Overfitted Learner*)**.** An overfitted learner f_k is a learner that is not underfitted, such that $Error_{test}(f_k) > Error_{train}(f_k)$.

Empirically, we can add a slack variable to relax the condition that the train error should be lower than the test error (e.g., require that the difference between them is greater than a value ϵ).

Definition 4 (*Limited Learner*)**.** A limited learner is an overfitted learner that is better than a random predictor in new samples.

Notice that, limited learners have a lower generalization error than dummy learners by construction since they are better than a random predictor.

The only base learners in the state-of-the-art that can be considered as limited learners are the *DT*s used to built a *RF*. In the *RF*, the maximum size of the *DT*s is not limited and they are not pruned. Thus, these *DT*s are overfitted by construction [5]. Hence, *DT*s in a *RF* are particular cases of limited learners that overfit as much as possible.



(a) $\lambda = 0.5$ (b) $\lambda = 1$

(c) $\lambda = 3$ (d) $\lambda = 5$

Fig. 2. Contour lines of the minimum overfitting objective function from different values of λ parameter. The scaler color represents the value of the function, closer to zero is better. The feasible region of the optimization problem is delimited to the right of the dotted red line.

4. Minimally Overfitted Ensemble

This Section proposes an ensemble learning framework, the *Minimally Overfitted Ensemble* (*MOE*), based on the concepts of a minimal overfitting model and limited learners. The *MOE* relies on the idea that assembling slightly overfitted limited learners will increase the performance of the ensemble.

A common way for choosing a well-fitted model is by defining a heuristic that identifies the minimum test error subject to a minimal difference between train error and test error. Thus, the overfitting is avoided in both the train and the test sets at the same time. Another way is to minimize the average error across different test sets by using techniques such as *k-fold cross-validation* [36,37]. In practice, there is no silver bullet to find the best *Machine Learning* (*ML*) model because the theoretical error is almost always unknown. Nevertheless, it is universally accepted that the solution can be founded by the analysis of the different kind of errors.

Following this approach, the minimum overfitting for binary classification can be defined in terms of the train and test errors. Since the objective is to promote minimum overfitting, the simplest approach is to minimize a loss function that considers the train and test error. In this function, the training error is the main contribution and the test error will be used to control the level of overfitting, as long as the test error is larger than the training error.

Definition 5 (*Minimally Overfitted Model*)**.** Given a set of *ML* models $\{\mathcal{F}\}$ fitted on the same train set, and evaluated on the same test set, the minimal overfitted model is the solution to:

$$\begin{aligned} &\arg\min_{f\in\mathcal{F}} L_{mo}(\lambda, f) = Error_{train}(f) + \lambda(Error_{test}(f) - Error_{train}(f))^2 \\ &\quad \text{subject to} \\ &\qquad Error_{test}(f) > Error_{train}(f) \end{aligned} \tag{1}$$

such that $\lambda \in (0, \infty)$.

Note that $L_{mo}(\lambda, f)$ is the overfitting loss function subject to $Error_{test}(f) > Error_{train}(f)$. The f model that minimizes L_{mo}, for a fixed λ, is the minimally overfitted among $\mathcal{F}$ set. The λ parameter allows controlling the level of overfitting that is considered to be the minimum. Values closer to zero promote overfitted models, whereas high values penalize the overfitting. The effects of the λ parameter are depicted in Fig. 2. The overfitting grade decreases as the value of λ increases. It is a subjective question what is the grade of overfitting that corresponds to the minimum overfitting. Therefore, the value of λ has to be set according to the specific problem. The empirical values of λ will mostly be a few units or at most tens of units, but not close to zero, and the cardinal of $\{\mathcal{F}\}$ will be large enough. Notice that the smaller the learning set, the fewer the possible different models there will be in $\{\mathcal{F}\}$.

Algorithm 1 presents the pseudo-code of the *MOE* framework. As in *Bagging*, the proposed framework begins by drawing m samples from the learning set with a sampling method S. Then, a base learner for each generated sample $\mathcal{L}_k^{(S)}$ is trained, and the minimally overfitted learner is selected. Finally, the prediction combination procedure is performed using the majority-vote rule. Notice that the *MOE* incorporates an additional step that includes limitations over the selected learners. In *Bagging*, the hyperparameters are the same for every individual learner. However, in the *MOE*, given a hyperparameter pool $\mathcal{H}$, the hyperparameter setting that provides the minimal overfitted learner is chosen. Therefore, for each sample $\mathcal{L}_k^{(S)}$, the method provides the minimal overfitted limited learner according to Eq. (1) (i.e., m limited learners). The computational time of the *MOE* is $O(m \cdot card(\mathcal{H}) \cdot (t + p_1 + p_2))$, where t is the complexity of training the base learner, p_1 is the complexity of predicting $\mathcal{L}_k^{(S)}$ with the base learner, p_2 is the complexity of predicting the *OOB* observations, and $card(\mathcal{H})$ is the size of the hyperparameter pool.

It is straightforward to show that the learners constructed according to the Definition 5 meet the limited learner conditions. Assuming that λ is set to the values recommended above and the hyperparameter pool $\mathcal{H}$ is large enough for the sample $\mathcal{L}_k^{(S)}$, then a subset of the learners will overfit. Among these overfitted learners, the minimally overfitted limited learner is selected

Algorithm 1 Minimally Overfitted Ensemble

Input: $(\mathcal{L}, \mathcal{H}, \lambda, n, m, S, BaseModel)$ ▷ S is the sampling method
$\mathbb{L} \leftarrow \emptyset$ ▷ initialize set of limited learners
for $k \in [1, \ldots, m]$ **do**
  $\mathcal{L}_k = s(\mathcal{L}, n)$
  $test = \mathcal{L} - \mathcal{L}_k^{(S)}$
  $L^* = \infty$
  for $h_j \in \mathcal{H}$ **do**
    $l(h_j) \leftarrow FitModel(h_j, \mathcal{L}_k)$ ▷ fit *jth* limited learner
    if $L_{mo}(\lambda, l(h_j)) < L^*$ **then**
      $L^* \leftarrow L_{mo}(\lambda, l(h_j))$
      $\mathbb{L}_k \leftarrow l(h_j)$ ▷ update *ith* limited learner
Output: $F(x) = argmax \sum_{k=1}^{m} \mathbb{L}_k(x)$ ▷ output as majority-voting rule

according to Definition 5. Notice that the training error of the limited learned is lower than the testing error and the underfitting is avoided because both error are minimized. Therefore, it is straightforward to show that the minimal overfitted learner outperforms the dummy learner. The corresponding theoretical bound error analysis is provided in Appendix. This error can be estimated in terms of the limited learners' accuracy and the correlation between them. In order to compute the minimum overfitting objective function, the $Error_{train}$ is calculated in the sample $\mathcal{L}_k^{(S)}$ and the $Error_{test}$ is calculated in the set $\mathcal{L} - \mathcal{L}_k^{(S)}$. The *OOB* approach for measuring the error provides a precise way to assess the overfitting level of each learner. Note that any evaluation metric could be used to calculate both errors. However, in some classification problems, the accuracy encourages overfitting over other evaluation measures such as *F1-score* or *Matthews correlation coefficient* [38].

As previously mentioned, a sampling method to obtain each $\mathcal{L}_k$ sample is needed. Given its similarities to *Bagging*, the first option for sampling is *Bootstrapping*. However, *Bootstrap* does not generate enough diversity for the minimum overfitting condition. Notice that re-balancing methods are very useful to avoid overfitting [34]. Here, the main objective is to guarantee the diversity of the sample to ensure that sufficiently different decision functions are combined. Thus, modifying the sampling proportion in each class will help reach diversity in the limited learners' decision functions. Next, a new sampling method called *Weighted RAndom Bootstrap* (*WRAB*), which leads to unbalanced samples and facilitates enough overfitting, is proposed to address this problem. Hence, this method will enable the combination of stable methods such as *SVM* or *kNN*.

4.1. Weighted Random Bootstrap

WRAB is a sampling procedure that consists on adding a layer of random weights that unbalance the sample size among the different classes in a classification problem. Let consider a sample size n and a learning set $\mathcal{L} = \{(\boldsymbol{x}_i, y_i) : i = 1, 2, \ldots, n\}$ where $\boldsymbol{x_i} \in \mathbb{R}^r$ and $y_i \in \mathbb{N}^m$. The procedure draws S samples from $\mathcal{L}$. Each *WRAB* sample is denoted by

$$\bigcup_{j=1}^{m} \mathcal{L}^{*b} = \{(\boldsymbol{x}_i^{*b}, y_i^{*b}) : i = 1, 2, \ldots, n_j\} \tag{2}$$

where $b = 1, 2, \ldots, S$ and $n_j \in [0, n]$. Each n_j ensures that $\sum_{j=1}^{m} w_j n_j = n$ where $\sum_{j=1}^{m} w_j = 1$ and $w_j \sim U(0, 1)$. Notice that some samples could appear several times because each *WRAB* sample is obtained by sampling with replacement.

Thus, the *MOE* framework has four hyperparameters: the number of samples for each base learner, the size of these samples, the grade of overfitting, and the sampling method (*WRAB* or *Bootstrap*). The first two parameters are widely known in Generative ensembles, while the last two are internal hyperparameters of the *MOE*.

5. Experiments

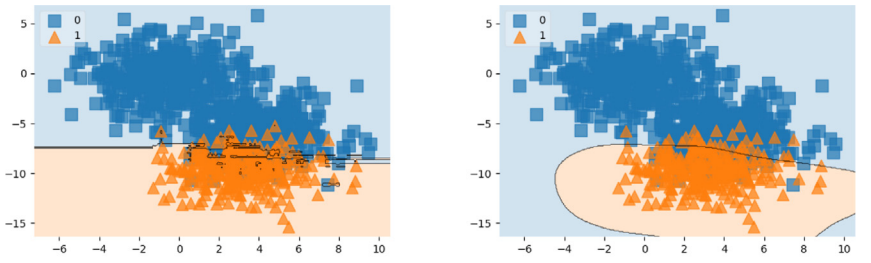
In this section, the proposed framework is studied and evaluated against several alternatives. First, the performance and hyperparameters of the *MOE* approach are discussed in an artificial dataset. Then, twenty-five binary classification datasets, with a variety of instances, number of features, and balance between the classes, are used to evaluate the proposal. Finally, the lessons learned are summarized. The source code and the data to reproduce the experiments can be found at https://github.com/URJCDSLab/moe.

5.1. Analysis of the overfitting hyperparameters in a controlled scenario

Let us consider a binary classification problem with three bivariate Normal distributions which centers are randomly placed at $(-0.65, -0.1)$, $(2.96, -9.52)$ and $(4.05, -6.06)$ in the square region $[-10, 10] \times [-10, 10] \in \mathbb{R}^2$ with a standard deviation equals to 2. The first two belong to the class 0 and the last belongs to 1. They are independent between them and contain 333 points. In this example, *RF* and *SVM* are considered as baseline *ML* models. Both models present similar classification errors (around 10%), obtained with 10-fold cross-validation by the sum of relative false positives and relative false negatives in 10 test partitions. The decision functions taken as a reference, estimated with *RF* and *SVM*, are presented in Fig. 3. In this case, for the *MOE* framework, the limited learners will use *SVM* with *Gaussian kernel*. In addition, the search space to minimize the minimal overfitting function is conformed with the regularization parameter $C \in \{1, 10, 100, 1000\}$, and the kernel coefficient $\gamma \in \{0.01, 0.1, 1, 10\}$. The results for the three different values of the overfitting parameter λ are shown in Fig. 4. For each example, 10 limited learners are built using *Bootstrap* where the sample size is equal to the 10% of the whole learning set. A proper definition of the objective function in the definition of minimal overfitted model is essential to achieve a high performance. A low value of λ implies an extremely overfitted method as shown in 4(a). In this case, the error rate equals 15% (50% worse than expected). The highest value of λ parameter achieves the expected 10% of error rate, but as shown in Fig. 4(c), the corresponding decision function is quite similar to the base algorithm. In fact, this situation would not necessarily be wrong, because the *SVM* decision function is better suited for one of the classes, as opposed to the *RF* decision function. As shown in Fig. 3, both decision functions (*RF* and *SVM*) are appropriate for this problem. An intermediate solution can be found in Fig. 4 for the largest value of λ. In this example, the distribution is denser near the mean in each of the three bivariate Normal clouds. Thus, low values of λ combined with *Bootstrap* sampling and a stable base algorithm lead to a lower diversity of the limited learners. Hence, to increase the diversity of the local decision functions for each learner, the λ parameter has to be increased resulting in high flexibility of the ensemble, as depicted in Fig. 4.

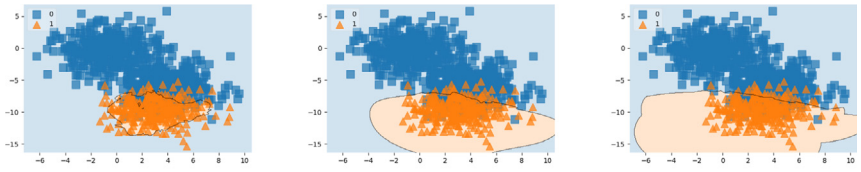
The underlying distribution of the data is usually unknown and scenarios like the one in the example are not uncommon, but difficult to confirm. It makes it difficult to adjust the overfitting parameter together with the number of limited learners and the sample size. The two last would be the lowest as possible that cover all the learning set. One way to solve this issue is to change the sampling method.

Fig. 5 shows the decision surfaces by using *WRAB* in the *MOE* framework (instead of *Boosting*). The impact of the parameter setting on the decision function decreases. Furthermore, none of the three results are quite similar to the *SVM* decision function (see Fig. 3(b)), especially as concerns the limit of overlap between



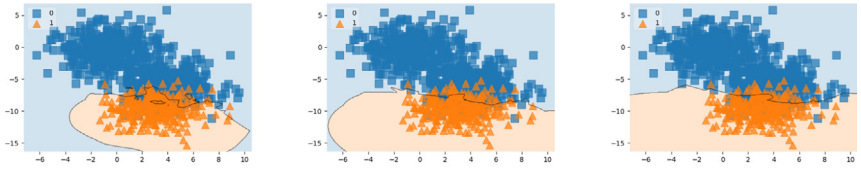
(a) Decision function estimated with RF (b) Decision function estimated with SVM

Fig. 3. Distribution of data for the example with 666 points for the blue class and 333 points in the orange class. It is shown the difference between the optimal decision surface for *RF* and *SVM*.



(a) $\lambda = 1$ (b) $\lambda = 3$ (c) $\lambda = 5$

Fig. 4. The three different decision surfaces for different values of the overfitting parameter in the *MOE* framework with bootstrap sampling instead of *WRAB*.



(a) $\lambda = 1$ (b) $\lambda = 3$ (c) $\lambda = 5$

Fig. 5. The three different decision surfaces for different values of the overfitting parameter in the MOE framework with *WRAB* sampling instead of bootstrap sampling.

Table 2
Detail of the errors for the *MOE* framework with different settings of sampling methodology and overfitting control parameter. The average (and standard deviation) of the limited learners for each setting is shown.

Hyperparameters	Train error	Test error	L_{mo}
bootstrap, $\lambda = 1$	0.00 (0.00)	0.19 (0.03)	0.04 (0.01)
bootstrap, $\lambda = 3$	0.04 (0.02)	0.13 (0.04)	0.07 (0.03)
bootstrap, $\lambda = 5$	0.07 (0.03)	0.12 (0.03)	0.08 (0.03)
WRAB, $\lambda = 1$	0.02 (0.03)	0.20 (0.03)	0.05 (0.02)
WRAB, $\lambda = 3$	0.02 (0.03)	0.16 (0.04)	0.05 (0.02)
WRAB, $\lambda = 5$	0.05 (0.03)	0.16 (0.04)	0.10 (0.03)

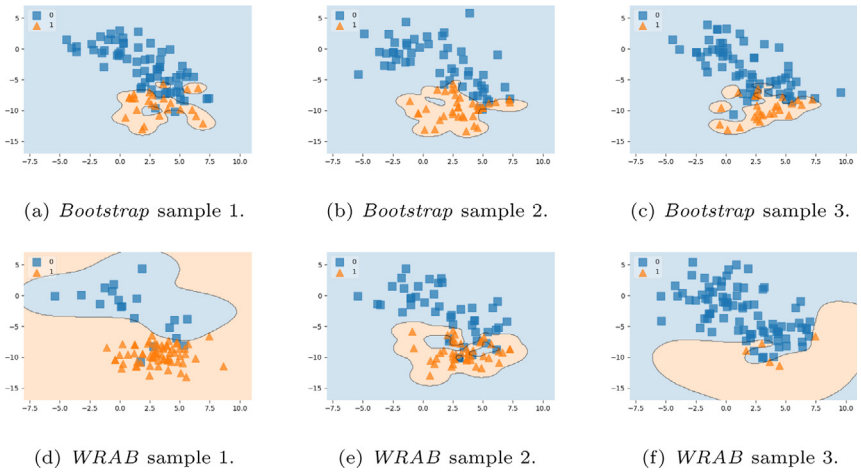
each class. In *Bootstrap*, it has been shown that low λ values tends to generate similar decision functions fitted around the densest region, like in the example for the orange class. Nevertheless, the unbalance of the sampling, provided by the *WRAB*, makes it easier to promote the overfitting in different regions of the learning set. Fig. 6 presents the decision functions obtained for different examples using both, *Bootstrap* and *WRAB* methods. As expected, it is shown that the *WRAB* method implies more diverse decision functions.

Table 2 presents the train and test errors of the *MOE* scheme for different sampling methods and different values for the λ parameter. Furthermore, the minimum overfitting loss function value is also available. The results of this table reveal that, for some elections of λ, the overfitting of the *WRAB* sampling is higher than that achieved with the *Bootstrap* methodology. Thus, the *WRAB* provides more diversity among samples, and keeps the minimum overfitting as well as the *Bootstrap* sampling.

5.2. Performance analysis of MOE variations

Next, the *MOE* framework is evaluated on a battery of binary classification datasets. The datasets have been obtained from the UCI repository [39], Penn Machine Learning Benchmarks [40], and LIBSVM Data [41]. A summary of its main features is presented in Table 3. In this experiment, we compare the *RF*, *SVM* with *Radial Basis Function kernel*, *kNN*, *XGBoost*, *GB*, and *ET* methods with three *MOE* variations using *DT*, *kNN*, and *SVM* as base learners. The predictive quality of the models is evaluated using the *Mathews Correlation Coefficient* (*MCC*) measure, which is one of the most popular evaluation metrics in the *ML* community [42], particularly for unbalanced datasets. Remark that the proportion of the minority class in some datasets is less than 0.30. For the sake of interpretability, *MCC* has been re-scaled to take values between 0 and 1. The hyperparameters are optimized using grid-search with 10-fold cross-validation. The hyperparameter grid is detailed in Table 4 (if a parameter is not stated, then the default value is selected).

The average and standard deviation of the *MCC* score in the test partitions of the 10-fold cross-validation for the best hyperparameters combination is presented in Table 5. The overall results show that the *MOE-DT* and the *MOE-KNN* were the best methods in 8 out of 25 datasets, being the *MOE-DT* the best on



(a) *Bootstrap* sample 1. (b) *Bootstrap* sample 2. (c) *Bootstrap* sample 3.

(d) *WRAB* sample 1. (e) *WRAB* sample 2. (f) *WRAB* sample 3.

Fig. 6. Different examples of samples with *WRAB* and *Bootstrap* for the artificial example if $\lambda = 1$. The *WRAB* leads to more different decision functions than *Bootstrap* in the case of *SVM*.

Table 3
Detail of the binary classification real datasets. The number of instances results after removing those that have missing values.

Dataset	N. Instances	N. Features	Minority class proportion
appendicitis	106	8	0.198
australian	690	15	0.445
backache	180	32	0.139
banknote	1372	5	0.445
breastcancer	569	31	0.373
bupa	345	6	0.490
cleve	303	14	0.455
colon-cancer	62	2001	0.355
diabetes	768	9	0.349
flare	1066	11	0.171
fourclass	862	3	0.356
german_numer	1000	25	0.300
haberman	306	4	0.265
heart	270	14	0.444
ilpd	579	11	0.285
ionosphere	351	35	0.359
kr_vs_kp	3196	37	0.478
liver-disorders	145	6	0.379
lupus	87	4	0.402
mammographic	830	5	0.486
mushroom	8124	23	0.482
r2	116	10	0.448
svmguide1	3089	5	0.353
svmguide3	1243	23	0.238
transfusion	748	5	0.238

MCC average. Notice that the standard deviation in all the methods is quite similar, so we can compare the mean performances in general. The Sign Test and the Wilcoxon Signed Rank Test [43] are employed for overall dataset pairwise comparison to assess the individual performance of each *MOE* variant. Concerning the Sign Test, the null hypothesis is that the *MOE* variant being studied is equivalent to the method it is compared to, and a rejection of the null hypothesis is that the *MOE* variant outperforms the alternative method. To reject the null hypothesis the number of wins needs to be greater or equal to the critical value. In the case of ties, they are split between the two methods. Fig. 7 (based on [44]) presents the performance of the *MOE* variations in terms of wins, ties and losses.

Particularly, the *MOE-DT* outperforms *RF*, *SVM*, *kNN*, *XGBoost*, and *ET*, with statistical confidence using $\alpha = 0.05$, and it is not guaranteed to be better than the *MOE-SVM*, the *MOE-KNN* and *GB*. The *MOE-KNN* outperforms *kNN* with statistical confidence using $\alpha = 0.05$ and, *RF* with statistical confidence using $\alpha = 0.10$. However, it is not guaranteed to be better than the *MOE-SVM*, the *MOE-DT*, *SVM*, *XGBoost*, *GB*, and *ET*. Finally, the *MOE-SVM* outperforms *RF*, *SVM*, *kNN* and *ET* with statistical confidence using $\alpha = 0.05$ and, *XGBoost* and *GB* with statistical confidence using $\alpha = 0.10$, and it is not guaranteed to be better than the *MOE-KNN*, the *MOE-DT*. Finally, using the Wilcoxon Signed Rank Test, with $\alpha = 0.05$, it is confirmed that the *MOE-DT* outperforms the other methods, with the exception of the *MOE-KNN* and the *MOE-SVM*. The *MOE-SVM* outperforms the other methods, with the exception of the *MOE-KNN*, the *MOE-SVM*, and *GB*. Moreover, the *MOE-SVM* only outperforms the *kNN*.

Globally, the *MOE* variations have a good performance in the experiments. This fact confirms the correct construction of the limited learners in terms of accuracy and diversity. The *MOE-SVM* and the *MOE-KNN* beat the single version of the algorithms in almost all the examples, and the *MOE-DT* beats the *RF* which also uses limited learners according to Definition 4. Besides, the *MOE-DT* wins more clearly over the rest. That apparent advantage of the *MOE-DT* could be produced by the *DT*s' overfitting propensity, along with their instability. Thus, *DT*'s provide both diversity and accuracy among all the limited learners. Those appreciations are derived from the best hyperparameters selected for each different *MOE* version, which can be found in Table 6.

Notice that the *WRAB* sampling is selected 64% of the time in the *MOE-DT*, against 68% and 56% in the *MOE-SVM* and the *MOE-kNN*, respectively. Therefore, for a stable model, the *WRAB* sampling is an excellent improvement on the desired diversity, but it also works well with unstable models like *DT*s. The selected λ values present two tendencies. The *MOE-kNN* and the *MOE-SVM* show a predilection for the highest value. The ensemble of minimally overfitted *SVM*s and *kNN*s benefit from less overfitting in general. However, the *MOE-DT* presents an inclination for the lowest value. There is no clear best value for the three *MOE* variants between 20 and 30, but these are preferred over 10. Notice that, as the complexity of the problem increases, more learners are needed to cover the whole learning set. Finally, the *MOE-kNN* tends not to select 0.20, with no differences between 0.10 and 0.30. The *MOE-DT* selects slightly more 0.30, and the *MOE-SVM* selects slightly more 0.10, in neither case is it a very substantial difference.

Table 4
ML methods used and their selected hyperparameter values.

Classification algorithms	Hyperparameters	Values
MOE	Wrab	True, False
	λ	1, 3, 5
	P. Sample	0.10, 0.30, 0.50
	N. Learners	10, 20, 30
KNN	k	1, 3, …, 11
SVM	C	1, 10, 100, 1000
	gamma	0.0001, 0.001, 0.01, 0.1, 1, 10
Random Forest	Max. Features	None, sqrt, log2
	N. Estimators	100, 300, 500
XGBoost	Max. Depth	3, 5, 7
	Eta	0.1, 0.2, 0.3
	N. Estimators	100, 300, 500
Extra Trees	Max. Features	None, sqrt, log2
	N. Estimators	100, 300, 500
Gradient Boosting	Max. Depth	3, 5, 7
	Learning Rate	0.1, 0.2, 0.3
	N. Estimators	100, 300, 500
	Max. Features	None, sqrt, log2

Table 5
The mean of the *MCC* scaled, standard deviation between parenthesis, calculated in each cross-validation test partition of the nine compared methods. Best results in bold for each dataset.

	MOE SVM	MOE KNN	MOE DT	Random forest	SVM	KNN	XGBoost	Gradient boosting	Extra trees
appendicitis	0.788 (0.177)	0.810 (0.146)	**0.818 (0.161)**	0.796 (0.150)	0.810 (0.146)	0.798 (0.135)	0.791 (0.148)	0.801 (0.157)	0.796 (0.150)
australian	0.871 (0.027)	0.862 (0.048)	0.881 (0.031)	0.871 (0.034)	0.871 (0.029)	0.843 (0.061)	**0.885 (0.022)**	0.876 (0.032)	0.865 (0.039)
backache	0.500 (0.000)	**0.653 (0.123)**	0.626 (0.139)	0.575 (0.163)	0.600 (0.147)	0.583 (0.143)	0.607 (0.154)	0.607 (0.170)	0.609 (0.162)
banknote	**1.000 (0.000)**	0.999 (0.002)	0.992 (0.008)	0.993 (0.005)	**1.000 (0.000)**	0.999 (0.003)	0.999 (0.003)	0.999 (0.003)	0.999 (0.002)
breastcancer	**0.982 (0.018)**	0.971 (0.017)	0.961 (0.024)	0.963 (0.019)	0.978 (0.018)	0.970 (0.019)	0.972 (0.021)	0.974 (0.012)	0.974 (0.009)
bupa	0.656 (0.109)	**0.661 (0.049)**	0.640 (0.069)	0.591 (0.101)	0.645 (0.091)	0.612 (0.110)	0.607 (0.094)	0.606 (0.093)	0.583 (0.080)
cleve	0.830 (0.067)	**0.838 (0.078)**	0.822 (0.046)	0.827 (0.060)	0.813 (0.063)	0.832 (0.073)	0.834 (0.034)	0.830 (0.043)	0.832 (0.070)
colon-cancer	0.876 (0.086)	**0.913 (0.092)**	0.886 (0.082)	0.877 (0.086)	0.862 (0.076)	0.775 (0.151)	0.859 (0.077)	0.877 (0.086)	0.887 (0.076)
diabetes	0.748 (0.034)	0.749 (0.033)	**0.751 (0.045)**	0.747 (0.033)	0.746 (0.041)	0.707 (0.034)	0.731 (0.037)	0.746 (0.046)	0.747 (0.036)
flare	0.639 (0.058)	**0.662 (0.045)**	0.658 (0.055)	0.617 (0.068)	0.613 (0.085)	0.611 (0.062)	0.632 (0.069)	0.633 (0.058)	0.586 (0.066)
fourclass	**1.000 (0.000)**	**1.000 (0.000)**	0.995 (0.008)	0.996 (0.006)	**1.000 (0.000)**	**1.000 (0.000)**	0.991 (0.010)	0.996 (0.008)	**1.000 (0.000)**
german_numer	0.719 (0.032)	0.688 (0.038)	**0.720 (0.044)**	0.716 (0.033)	0.713 (0.027)	0.633 (0.042)	0.715 (0.050)	0.715 (0.048)	0.692 (0.046)
haberman	0.648 (0.123)	0.632 (0.102)	**0.653 (0.119)**	0.583 (0.078)	0.595 (0.089)	0.625 (0.089)	0.580 (0.110)	0.576 (0.091)	0.571 (0.059)
heart	**0.868 (0.056)**	0.861 (0.066)	0.847 (0.049)	0.837 (0.064)	0.854 (0.059)	0.834 (0.070)	0.814 (0.055)	0.823 (0.052)	0.846 (0.058)
ilpd	0.662 (0.062)	0.643 (0.063)	**0.665 (0.040)**	0.628 (0.049)	0.626 (0.055)	0.625 (0.086)	0.622 (0.068)	0.655 (0.073)	0.658 (0.041)
ionosphere	0.954 (0.042)	0.899 (0.059)	0.941 (0.051)	0.929 (0.039)	**0.955 (0.056)**	0.870 (0.050)	0.924 (0.052)	0.943 (0.041)	0.939 (0.039)
kr_vs_kp	0.993 (0.005)	0.947 (0.014)	0.994 (0.003)	0.995 (0.005)	0.994 (0.004)	0.953 (0.014)	0.995 (0.004)	**0.997 (0.004)**	0.996 (0.004)
liver-disorders	0.748 (0.058)	0.752 (0.055)	0.780 (0.105)	0.758 (0.097)	0.724 (0.084)	0.715 (0.133)	0.777 (0.102)	**0.785 (0.088)**	0.757 (0.054)
lupus	0.770 (0.123)	0.757 (0.190)	**0.780 (0.116)**	0.744 (0.198)	0.743 (0.126)	0.742 (0.178)	0.712 (0.160)	0.716 (0.152)	0.709 (0.208)
mammographic	0.815 (0.040)	0.814 (0.032)	**0.821 (0.035)**	0.789 (0.023)	0.810 (0.036)	0.802 (0.028)	0.813 (0.035)	0.807 (0.037)	0.756 (0.042)
mushroom	**1.000 (0.000)**	**1.000 (0.000)**	**1.000 (0.000)**	**1.000 (0.000)**	**1.000 (0.000)**	**1.000 (0.000)**	**1.000 (0.000)**	**1.000 (0.000)**	**1.000 (0.000)**
r2	**0.798 (0.133)**	0.749 (0.104)	0.781 (0.117)	0.765 (0.105)	0.770 (0.150)	0.722 (0.170)	0.756 (0.105)	0.785 (0.087)	0.784 (0.133)
svmguide1	0.964 (0.011)	0.959 (0.009)	0.968 (0.008)	0.966 (0.010)	0.968 (0.009)	0.960 (0.007)	**0.969 (0.010)**	0.971 (0.011)	0.966 (0.008)
svmguide3	0.764 (0.054)	0.694 (0.056)	0.775 (0.049)	0.766 (0.060)	0.767 (0.043)	0.708 (0.056)	**0.796 (0.051)**	0.785 (0.048)	0.758 (0.050)
transfusion	0.684 (0.051)	**0.686 (0.061)**	0.683 (0.056)	0.626 (0.074)	0.676 (0.048)	0.662 (0.046)	0.644 (0.043)	0.651 (0.043)	0.602 (0.078)
Mean	0.811 (0.055)	0.808 (0.059)	**0.818 (0.058)**	0.798 (0.062)	0.805 (0.059)	0.783 (0.070)	0.801 (0.061)	0.806 (0.059)	0.796 (0.060)

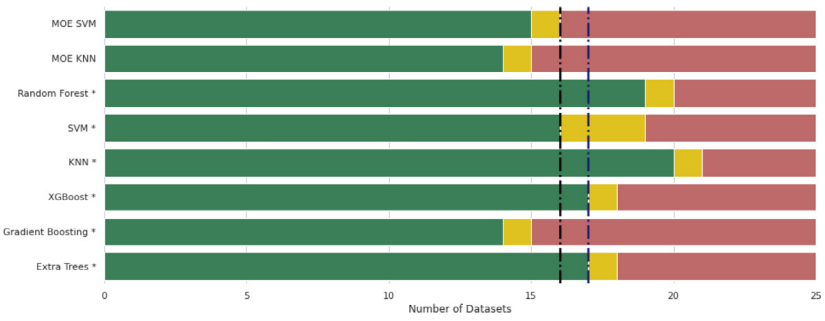
5.3. Lessons learned

This section synthesizes the main lessons learned from the proposal in the case studies.

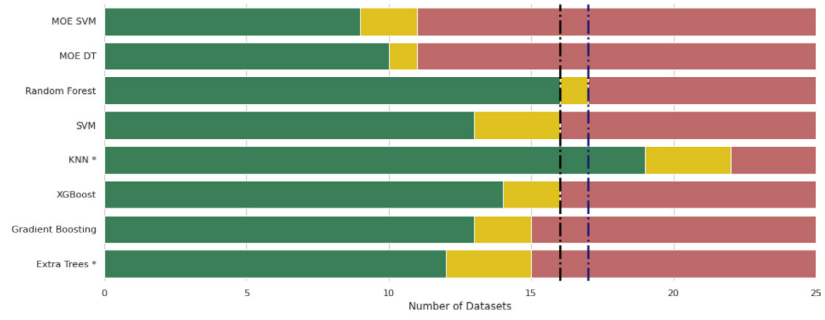
One of the strengths of Generative ensembles based on sampling lies on the diversity of the different decision regions generated by the base learners fitted on each sample. If enough base learners are used on a variety of samples, covering the whole learning set with a high diversity, the resulting combination of predictions will outperform the individual result. Every learner has to ensure sufficient accuracy.

The diversity of the decision regions could be reached using re-sampling techniques. However, in stable algorithms, the perturbations in the learning set generated using re-sampling methods such as *Bootstrap* are not enough to significantly change the resultant decision functions. It is possible to generate more diverse samples in classification problems by randomly sampling over the labels, and as a result, generating more diverse decision functions.

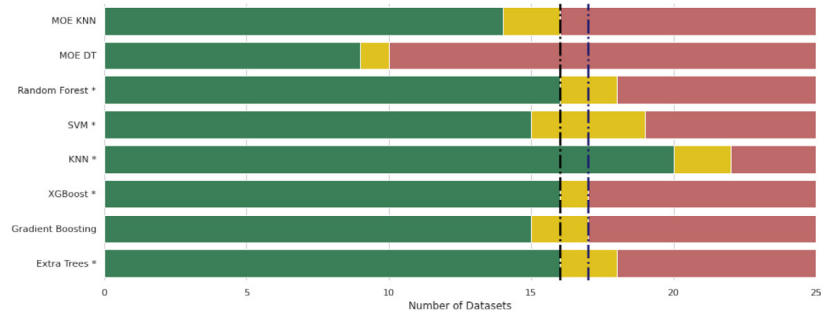
The *WRAB* sampling method has been developed to assemble any minimally overfitted classifier for binary problems in the *MOE* framework. A Wilcoxon test is performed to rigorously evaluate the differences between *WRAB* and *bootstrap*. We obtain the best score for *WRAB* and *bootstrap* for the 25 datasets. To test the null hypothesis that there is no performance difference, we apply the two-sided test, which provides a p-value $= 0.0092$. Hence,



(a) *MOE-DT*.



(b) *MOE-KNN*.



(c) *MOE-SVM*.

Fig. 7. Performance of each *MOE* variant in terms of wins (green), ties (yellow), and losses (red). The vertical dashed lines represent the critical values, 16 and 17 for two different confidence levels, $\alpha = \{0.05, 0.10\}$, respectively. The methods marked with * represent those where the corresponding MOE variant has significantly outperformed them using the Wilcoxon Signed Rank Test ($\alpha = 0.05$).

Table 6
Frequency of the best hyperparameters in the *MOE* framework.

Hyperparameter	Value	MOE-SVM	MOE-DT	MOE-kNN
Wrab	False	8	9	11
	True	17	16	14
λ	1	6	15	7
	3	6	5	5
	5	13	5	13
N. Learners	10	5	3	6
	20	9	11	9
	30	11	11	10
P. Sample	0.10	10	7	10
	0.20	8	7	5
	0.50	7	11	10

the null hypothesis can be rejected at a confidence level of 95%, concluding that there is a difference in performance between the two methods. To confirm that the median of the differences can be assumed to be positive, we use the one-side test, obtaining a p-value $= 0.0046$. This shows that the null hypothesis that the median is negative can be rejected at a confidence level of 95% in favor of the alternative that the median is greater than zero. Thus, it can be concluded that the *WRAB* reaches better results than *bootstrap* in this context.

Regarding the hyperparameters of the *MOE*, it is recommendable following the experimental results to use the *WRAB* and 30 learners as default values. In most cases, these values work well. As a starting point, the sample proportion must be set to 0.10 for the *MOE-SVM* and 0.50 for the *MOE-kNN* and the *MOE-DT*. The overfitting control parameter should be set to 1 for the *MOE-DT* and 5 for the *MOE-SVM* and the *MOE-kNN*. The general advice is to modify the number of learners and the proportion of the sample to lower values before changing the λ or the *WRAB* recommended values.

The minimum overfitting of the learners provides a good performance in the ensemble, as long as the samples cover most of the learning set. The perturbations obtained by re-sampling methods have more impact on a minimally overfitted learner's decision function than a well-fitted learner. Hence, the decision function will vary more in a minimally overfitted learner than in a well-fitted learner. Moreover, the sample size will be smaller than other ensembles like *RF*, because the diversity among the learners is higher and the learners are sufficiently accurate.

6. Conclusions and future work

In this paper, the notion of limited learner is introduced. A limited learner is an overfitted learner whose predictions are better than random predictions. Founded on this notion, a general re-sampling-based *ML* framework called *MOE* is proposed. It is designed to work with any base *ML* algorithm, both stable and unstable, overcoming the limitations of stable learners in ensembles. The key idea behind the *MOE* is a hyper-parameter search in each sample, where the learner is selected using the minimum overfitting concept, which promotes slightly overfitted models. The results are minimally overfitted learners, a specific type of limited learners. In addition, the datasets for training each single learner are generated using a new sampling method. The *WRAB* is a layer on top of the *Bootstrap* sampling method that modifies the original balance between the classes in a classification problem. Therefore, the resulting single learners are more diverse, especially when working with stable algorithms. The resultant learners built by the *MOE* framework are combined using a majority-voting rule.

The performance of the proposal was evaluated on eighteen real datasets with three different base algorithms: *SVM*, *DT*, and *kNN*. The *MOE* variations outperformed their single version algorithms counterpart. The best overall performance has been achieved by the *MOE-DT*. Furthermore, the hyperparameter analysis confirms that the instability and overfitting tendency of the base algorithm benefits the overall performance of the *MOE* framework. In stable base algorithms, the *WRAB* overcomes the limitation of lack of variability, generating diverse single learners using stable base algorithms such as *SVM* with Gaussian kernel and outperforming their single version. Further, the proposal sightly outperforms the reference limited learner ensemble method, *RF*.

Regarding the minimum overfitting, the proposed definition needs a pool of *ML* models, which causes an increase in the training time of the proposed method and limits experiments for large datasets. The runtimes of the *MOE* variations are driven by the exhaustive hyperparameter search that is performed on each sample to select the minimally overfitted learner according to Definition 5. Therefore, adding a large number of learners increases the training time considerably, especially when assembling very complex models such as *SVM*s. Future work will focus on narrowing down the pool of learners to select the minimally overfitted learner. The *WRAB* sampling method provides diversity in the ensemble when the base algorithm is stable, but it might not be enough in some cases. The sampling and the overfitting parameter could be context-aware to ensure the diversity of the limited learners and choose more or less stable *ML* models. For instance, by increasing the number of samples in complex regions or reducing it when using stable models in simple regions. A promising research line would be to explore the inclusion within the framework of novel approaches for the treatment of unbalanced data. Finally, the expansion of the *MOE* framework to multiclass classification and regression will be studied by revisiting the definition of the *Minimally Overfitted Model* and the *WRAB* to these *ML* paradigms.

CRediT authorship contribution statement

Víctor Aceña: Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing. **Isaac Martín de Diego:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing. **Rubén R. Fernández:** Writing – original draft, Writing – review & editing. **Javier M. Moguerza:** Conceptualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

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

Acknowledgments

This research has been supported by grants from the Madrid Autonomous Community (Ref: IND2019/TIC-17194) and the Spanish Ministry of Economy and Competitiveness under the Retos-Investigación program: MODAS-IN (Ref: RTI-2018-094269-B-I00); and donation of the Titan V GPU by NVIDIA Corporation. Special thanks to JOF ASSOCIATES INT S.L.U. Sincere gratitude to Aníbal R. Figueiras Vidal for his valuable advice in developing this work, without which it would not have the quality achieved.

Appendix. Upper bound for the generalization error

A MOE is a collection of minimally overfitted learners, which is a particular case of limited learners. Hence, a *MOE* is an ensemble of limited learners $l_k(\boldsymbol{x})$ where $\boldsymbol{x}$ is an input vector, and each limited learner votes with the same weight for the most suitable class at input $\boldsymbol{x}$. Every limited learner can be represented by a parametric function $l(\boldsymbol{x}, k)$ where k is a parameter vector defined in terms of the disjoint regions of input space. For binary classifiers, the parametric function takes the form:

$$l(\boldsymbol{x}, \theta_k) = \sum_{j=1}^{J} I_{[\boldsymbol{x}\in R_j]} \cdot \hat{y}_j$$

where R_j defines the jth disjoint region of the input space which is induced by the classifier and the y_j represent the label assigned to the jth region. For example, for a linear *SVM*, the separating hyperplane defined by $sign(\sum_{i\in sv}(\omega^T \boldsymbol{x}_i + b))$ generates two regions R_1 and R_2.

Hence, given the definition of the ensemble in terms of limited learners, the generalization error for MOE is parallel to Breiman's analysis [5] for *Random Forest*.

Given a collection of classifiers $l_1(\boldsymbol{x}), l_2(\boldsymbol{x}), \ldots, l_m(\boldsymbol{x})$, and the training set following the underlying distribution of the random vectors $\boldsymbol{X}, Y$. The classification margin is

$$m(\boldsymbol{X}, Y) = P_\Theta(l(\boldsymbol{X}, \Theta) = Y) - max_{c\neq Y} P_\Theta(l(\boldsymbol{X}, \Theta) = c) \tag{A.1}$$

The generalization error for a voting ensemble can be bounded by measuring the diversity and the accuracy of the learners. The accuracy is measured by the expected strength, and it is defined as

$$PE = P_{\boldsymbol{X},Y}(m(\boldsymbol{X}, Y) < 0) \tag{A.2}$$

In the balanced binary case, it can be formulated as

$$m(\boldsymbol{X}, Y) = 2P_\Theta(l(\boldsymbol{X}, \Theta) = Y) - 1 \tag{A.3}$$

and meeting the condition $\mu > 0$ means that $E_{\boldsymbol{X},Y}(P_\Theta(l(\boldsymbol{X}, \Theta) = Y)) > 0.5$ and, then, we reach the condition of a weak classifier. The limited learners meet this condition by Definition 4.

Therefore, if $m(\boldsymbol{X}, Y) > 0$, the collection votes for the right class. The generalization error for a limited learner ensemble is

$$PE = P_{\boldsymbol{X},Y}(m(\boldsymbol{X}, Y) < 0) \tag{A.4}$$

Applying Chebyshev's inequality with $\mu > 0$,

$$PE = \frac{var_{\boldsymbol{X},Y}(m(\boldsymbol{X}, Y))}{\mu^2} \tag{A.5}$$

Now, we are going to deal with the expression for the variance. Let

$$\hat{c} = argmax_{c \neq Y} P_{\Theta}(l(\boldsymbol{X}, \Theta) = c) \tag{A.6}$$

be the class with the most incorrect votes.

Then,

$$m(\boldsymbol{X}, Y) = P_{\Theta} l(\boldsymbol{X}, \Theta) - P_{\Theta}(l(\boldsymbol{X}, \Theta) = c) = E_{\Theta}(m^*(\boldsymbol{X}, Y, \Theta)) \tag{A.7}$$

where

$$m^*(\boldsymbol{X}, Y, \theta) = I_{[l(\boldsymbol{x},\theta)=Y]} - I_{[l(\boldsymbol{x},\theta)=\hat{c}]} \tag{A.8}$$

Assuming that Θ and Θ' are independent and identically distributed,

$$\begin{aligned} m(\boldsymbol{X}, Y)^2 &= E_{\Theta}(m^*(\boldsymbol{X}, Y, \Theta))^2 \\ &= E_{\Theta,\Theta'}(m^*(\boldsymbol{X}, Y, \Theta), m^*(\boldsymbol{X}, Y, \Theta')) \end{aligned} \tag{A.9}$$

Therefore,

$$\begin{aligned} var_{\boldsymbol{X},Y}(m(\boldsymbol{X}, Y)) &= E_{\Theta,\Theta'}(cov_{\boldsymbol{X},Y}(m^*(\boldsymbol{X}, Y, \Theta), m^*(\boldsymbol{X}, Y, \Theta'))) \\ &= E_{\Theta,\Theta'}(\rho(\Theta, \Theta')\sigma(\Theta)\sigma(\Theta')) \end{aligned} \tag{A.10}$$

where,

$$\rho(\theta, \theta') = corr_{\boldsymbol{X},Y}(m^*(\boldsymbol{X}, Y, \theta), m^*(\boldsymbol{X}, Y, \theta')) \tag{A.11}$$

is the correlation between two learners of the ensemble, for fixed θ and θ', and $\sigma(\theta)$ is the square-root of

$$\sigma^2(\theta) = var_{\boldsymbol{X},Y}(m^*(\boldsymbol{X}, Y, \theta)) \tag{A.12}$$

From (A.10) and the variance definition,

$$var_{X,Y}(m(X, Y)) = \overline{\rho} \cdot E_{\Theta}(\sigma(\Theta))^2 \leq \overline{\rho} \cdot E_{\Theta}(\sigma^2(\Theta)) \tag{A.13}$$

where

$$\overline{\rho} = \frac{E_{\Theta,\Theta'}(\rho(\Theta, \Theta')\sigma(\Theta)\sigma(\Theta'))}{E_{\Theta,\Theta'}(\sigma(\Theta)\sigma(\Theta'))} \tag{A.14}$$

is the average correlation between all learners taken in pairs.

Now, from (A.12)

$$E_{\Theta}(\sigma^2) = E_{\Theta}(E_{\boldsymbol{X},Y}(m^*(\boldsymbol{X}, Y, \theta')^2) - E_{\boldsymbol{X},Y}(m^*(\boldsymbol{X}, Y, \theta'))^2) \tag{A.15}$$

From (A.8) we know that $m^*(\boldsymbol{X}, Y, \Theta)^2 \leq 1$, thus

$$\begin{aligned} E_{\Theta}(E_{\boldsymbol{X},Y}(m^*(\boldsymbol{X}, Y, \Theta'))^2) &\geq (E_{\Theta}(E_{\boldsymbol{X},Y}(m^*(\boldsymbol{X}, Y, \Theta'))))^2 \\ &= (E_{\boldsymbol{X},Y}(E_{\Theta}(m^*(\boldsymbol{X}, Y, \Theta'))))^2 \\ &= (E_{\boldsymbol{X},Y}(m^*(\boldsymbol{X}, Y, \Theta')))2 \\ &= \mu^2 \end{aligned} \tag{A.16}$$

Finally,

$$E_{\Theta}(\sigma^2(\Theta)) \geq 1 - \mu^2 \tag{A.17}$$

Using (A.17) into (A.13), and the result into PE (A.5), we obtain the upper bound of the generalization error for a limited learner ensemble, in terms of accuracy and diversity, given by μ and $\overline{\rho}$:

$$PE^* \leq \frac{\overline{\rho}(1 - \mu^2)}{\mu^2} \tag{A.18}$$

References

[1] G. Valentini, Random aggregated and bagged ensembles of svms: an empirical bias–variance analysis, in: International Workshop on Multiple Classifier Systems, Springer, 2004, pp. 263–272.
[2] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[3] Y. Freund, R.E. Schapire, et al., Experiments with a new boosting algorithm, in: Machine Learning: Proceedings of the Thirteenth International Conference, ACM, 1996, pp. 148–156.
[4] R.E. Schapire, The strength of weak learnability, Mach. Learn. 5 (2) (1990) 197–227.
[5] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[6] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, et al., Xgboost: extreme gradient boosting, (4) 2015, R package version 0.4-2 1.
[7] K.M. Ting, J.R. Wells, S.C. Tan, S.W. Teng, G.I. Webb, Feature-subspace aggregating: ensembles for stable and unstable learners, Mach. Learn. 82 (3) (2011) 375–397.
[8] Z.-H. Zhou, Ensemble Methods: Foundations and Algorithms, CRC Press, 2012.
[9] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: Bagging, boosting, and variants, Mach. Learn. 36 (1) (1999) 105–139.
[10] S. Han, H. Kim, Y.-S. Lee, Double random forest, Mach. Learn. 109 (8) (2020) 1569–1586.
[11] M.J. Way, J.D. Scargle, K.M. Ali, A.N. Srivastava, Advances in Machine Learning and Data Mining for Astronomy, CRC Press, 2012.
[12] G. Valentini, F. Masulli, Ensembles of learning machines, in: Italian Workshop on Neural Nets, Springer, 2002, pp. 3–20.
[13] L. Lam, S. Suen, Application of majority voting to pattern recognition: an analysis of its behavior and performance, IEEE Trans. Syst. Man Cybern. A 27 (5) (1997) 553–568.
[14] M.P. Perrone, L.N. Cooper, When networks disagree: Ensemble methods for hybrid neural networks, Tech. rep., Brown Univ Providence Ri Inst for Brain and Neural Systems, 1992.
[15] L. Xu, A. Krzyzak, C.Y. Suen, Methods of combining multiple classifiers and their applications to handwriting recognition, IEEE Trans. Syst. Man Cybern. 22 (3) (1992) 418–435.
[16] D.G. Stork, R.O. Duda, P.E. Hart, D. Stork, Pattern Classification, Wiley-Interscience, 2001.
[17] D. Partridge, W.B. Yates, Engineering multiversion neural-net systems, Neural Comput. 8 (4) (1996) 869–893.
[18] M.F. Hassan, I. Abdel-Qader, B. Bazuin, A new method for ensemble combination based on adaptive decision making, Knowl.-Based Syst. 233 (2021) 107544.
[19] X. Yu, Q. Peng, L. Xu, F. Jiang, J. Du, D. Gong, A selective ensemble learning based two-sided cross-domain collaborative filtering algorithm, Inf. Process. Manage. 58 (6) (2021) 102691.
[20] X. Yu, Y. Chu, F. Jiang, Y. Guo, D. Gong, Svms classification based two-side cross domain collaborative filtering by inferring intrinsic user and item features, Knowl.-Based Syst. 141 (2018) 80–91.
[21] X. Yu, F. Jiang, J. Du, D. Gong, A cross-domain collaborative filtering algorithm with expanding user and item features via the latent factor space of auxiliary domains, Pattern Recognit. 94 (2019) 96–109.
[22] J.J. Rodriguez, L.I. Kuncheva, C.J. Alonso, Rotation forest: A new classifier ensemble method, IEEE Trans. Pattern Anal. Mach. Intell. 28 (10) (2006) 1619–1630.
[23] M.K. Titsias, A. Likas, Mixture of experts classification using a hierarchical mixture model, Neural Comput. 14 (9) (2002) 2221–2244.
[24] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Mach. Learn. 63 (1) (2006) 3–42.
[25] L.K. Hansen, P. Salamon, Neural network ensembles, IEEE Trans. Pattern Anal. Mach. Intell. 12 (10) (1990) 993–1001.
[26] L. Rokach, Ensemble Learning: Pattern Classification using Ensemble Methods, Vol. 85, World Scientific, 2019.
[27] S. González, S. García, J. Del Ser, L. Rokach, F. Herrera, A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities, Inf. Fusion 64 (2020) 205–237.
[28] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. System Sci. 55 (1) (1997) 119–139.
[29] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, Adv. Neural Inf. Process. Syst. 30 (2017) 3146–3154.
[30] T.G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization, Mach. Learn. 40 (2) (2000) 139–157.
[31] S.-J. Wang, A. Mathew, Y. Chen, L.-f. Xi, L. Ma, J. Lee, Empirical analysis of support vector machine ensemble classifiers, Expert Syst. Appl. 36 (3) (2009) 6466–6476.

[32] Y. Liu, X. Yu, J.X. Huang, A. An, Combining integrated sampling with svm ensembles for learning from imbalanced datasets, Inf. Process. Manage. 47 (4) (2011) 617–631.
[33] Q. Wang, Z. Luo, J. Huang, Y. Feng, Z. Liu, A novel ensemble method for imbalanced data learning: Bagging of extrapolation-smote svm, Comput. Intell. Neurosci. 2017 (2017).
[34] M. Lázaro, F. Herrera, A.R. Figueiras-Vidal, Ensembles of cost-diverse bayesian neural learners for imbalanced binary classification, Inform. Sci. 520 (2020) 31–45.
[35] Y. Zhang, G. Cao, B. Wang, X. Li, A novel ensemble method for k-nearest neighbor, Pattern Recognit. 85 (2019) 13–25.
[36] G. Wahba, et al., Support vector machines, reproducing kernel hilbert spaces and the randomized gacv, Adv. Kernel Methods-Support Vector Learn. 6 (1999) 69–87.
[37] G.H. Golub, M. Heath, G. Wahba, Generalized cross-validation as a method for choosing a good ridge parameter, Technometrics 21 (2) (1979) 215–223.
[38] M. Grandini, E. Bagli, G. Visani, Metrics for multi-class classification: an overview, 2020, arXiv preprint arXiv:2008.05756.
[39] D. Dua, C. Graff, [UCI] machine learning repository, 2017, http://archive.ics.uci.edu/ml.
[40] R.S. Olson, W. La Cava, P. Orzechowski, R.J. Urbanowicz, J.H. Moore, Pmlb: a large benchmark suite for machine learning evaluation and comparison, BioData Mining 10 (36) (2017) 1–13, http://dx.doi.org/10.1186/s13040-017-0154-4.
[41] C.-C. Chang, Libsvm data: Classification, regression, and multi-label, 2008, http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.
[42] A.R. Redondo, J. Navarro, R.R. Fernández, I.M. de Diego, J.M. Moguerza, J.J. Fernández-Muñoz, Unified performance measure for binary classification problems, in: International Conference on Intelligent Data Engineering and Automated Learning, Springer, 2020, pp. 104–112.
[43] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[44] D.V. Oliveira, G.D. Cavalcanti, R. Sabourin, Online pruning of base classifiers for dynamic ensemble selection, Pattern Recognit. 72 (2017) 44–58.