Applied Intelligence (2023) 53:4996–5012
https://doi.org/10.1007/s10489-022-03712-z

Boosted support vector machines with genetic selection

A. Ramirez-Morales[1] · J. U. Salmon-Gamboa[2] · Jin Li[1] · A. G. Sanchez-Reyna[3] · A. Palli-Valappil[1]

Accepted: 29 April 2022 / Published online: 17 June 2022

Abstract
This paper describes the experimental studies of ensembles of binary classifiers conformed of individual support vector machines. The GenBoost-SVM method is proposed to construct such ensembles. Our ensembles considered an adaptive boosting algorithm. We analyzed different pre-selections using genetic algorithms to reduce the size of the training samples and hence the training times. These genetic selections addressed the imbalanced data challenge directly. Furthermore, in our ensembles, diversity and early stopping were considered to help to reduce the generalization error. We proposed 56 different types of ensembles that permute the support vector machine kernels, genetic selections and diversity. We found that our ensembles, which consider genetic selections and diversity, exhibit competitive performances when compared with various popular classifiers, and for imbalanced data, they outperform most of the considered popular classifiers. We show that using different support vector machine kernels leads to enhanced performances. To the best of our knowledge, this is the first study that combines adaptive boosted ensembles, genetic selections, and support vector machines.

Keywords Support vector machines · Adaboost · Genetic selection · Ensemble

1 Introduction

Ensemble methods that create a strong classifier from a linear combination of weak classifiers are powerful tools to tackle real world machine learning challenges [1, 2]. Building an ensemble with boosting techniques has recently become popular; particularly, in an ensemble built using the adaptive boosting algorithm (AdaBoost), the generalization error is rapidly reduced when adding classifiers since the algorithm enhances/boosts the weights of misclassified samples for every added classifier in the ensemble. In supervised learning, the ensemble's base classifiers are trained with well-characterized data and are required to be accurate only slightly better than random guess (weak classifiers). The AdaBoost algorithm dictates how to combine these base classifiers into a single classifier and allows for properties of data not used in the training phase to be inferred [3–5]. Examples of classifiers that have been successfully combined with Adaboost are: artificial neural networks [6], decision trees [7], k-nearest neighbors [8], naive Bayes [9] and convolutional neural networks [10].

Support vector machines (SVM) [11] are expected to be useful base classifiers in an ensemble, since they offer high accuracy and work well in high dimensional spaces. Their decision functions can be easily tuned by appropriate kernel functions [12]. SVMs rely entirely on a geometric approach to the binary classification problem, making them robust against noisy data and outliers [13, 14]. The use of SVMs is wide in several fields and is in constant expansion; for example, in agricultural sciences [15], stock

✉ Jin Li
lijin@knu.ac.kr

A. Ramirez-Morales
andres@knu.ac.kr

J. U. Salmon-Gamboa
jorge.salmon@kcl.ac.uk

A. G. Sanchez-Reyna
ing.agsreyna19@gmail.com

A. Palli-Valappil
anusha.pv@btech.christuniversity.in

[1] Center for High Energy Physics, Kyungpook National University, 80 Daehak-ro, Daegu, 45166, Republic of Korea

[2] Department of Physics and London Centre for Nanotechnology, King's College London, Street, Strand, WC2R 2LS, London, United Kingdom

[3] Unidad Academica de Ingenieria Electrica, Universidad Autonoma de Zacatecas, Jardin Juarez 147, Zacatecas, 98000, Zacatecas, Mexico

market prediction [16], medical image recognition [17], swarm computing [18], health monitoring [19], and other disciplines.

An SVM classifier needs to be optimized in terms of its kernel function and hyper-parameters. Determining the optimal kernel and hyper-parameters is time consuming and computationally expensive. For example, the simplest practice is to perform a grid search to assess the performance of the SVM for several configurations. The number of configurations becomes intractable when the considered amount of kernel functions and hyper-parameters grows, thus rendering it impractical. To overcome this problem, the construction of ensembles using SVMs as base classifiers to avoid grid searches has been under study. Moreover, SVMs present a major inherent drawback: the training process escalates in time and memory complexity ($O(n^3)$ and $O(n^2)$, respectively) as the number of samples n in the training set increases, limiting the use of SVM within the realm of big data analysis. To reduce the number of training samples, and hence the training time, for an SVM to obtain a good generalization performance, a promising approach is to use genetic algorithm selections [20].

To collectively study the benefits above mentioned when using SVMs, this paper investigates a novel method called GenBoost-SVM that combines genetic algorithms (GAs), SVMs, and AdaBoost ensembles. We constructed SVMs ensembles with AdaBoost weights and selected the most suitable data samples utilizing a genetic algorithm. Particularly, we boosted the SVM classifiers by changing the kernel hyper-parameters for different kernels. Moreover, we varied the genetic selections using several scores and selection types to find a reduced data sample that potentially contains the vectors needed to find an effective SVM separating hyper-plane. We also included ensemble diversity selection criteria to address the accuracy-diversity dilemma. We found the best ensemble configuration and genetic selection that maintains a competitive classifier performance and reasonable training times. The latter is achieved since no SVM hyper-parameter tuning is needed and the number of training vectors is reduced.

This paper is organized as follows: Section 2 provides a description of the previous related work. Section 3 contains our proposed methodology and the manner in which the ensembles are constructed is found in Algorithm 2 GenBoost-SVM. Section 4 outlines the experimental details of our study. Section 5 presents and discusses our obtained results and gives a description of the best SVM ensembles found. We give a formal statiscal comparison of our ensembles against several popular classifiers (PCs). Our concluding remarks are given in Section 6.

2 Related work

Ensembles with SVMs as base classifiers have previously helped avoid the time consuming hyper-parameter grid searches. For example, X. Li et al. [21] showed that SVMs are suitable for use in ensembles that follow the AdaBoost algorithm and that they could improve the classification of imbalanced data sets without searching for optimal kernels and hyper-parameters. Moreover, H.C. Kim et al. [22] constructed SVM ensembles with bootstrap aggregation, demonstrating improvement with respect to single SVM in real data sets.

Genetic selections for training data reduction for support vector machines were studied by Nalepa and Kawulok [23]. In another work, they treated the problem of defining the genetic algorithm parameters with an adaptive selection [24]; they also implemented a memetic algorithm which uses the knowledge obtained during the genetic selection to improve the selected training data [25]. Verbiest et al. [26] demonstrated the effectiveness of using genetic algorithms to improve SVMs, by comparing the classifier accuracy between standard, adaptive, and steady genetic selections. Fernandes et al. [27] treated imbalanced data sets by constructing highly accurate and diverse ensembles. They utilized adaptive genetic sampling methods to optimize the SVMs that conform the ensemble. Kawulok et al. [28] combined the training data selection and the search of an optimal SVM kernel with the aid of genetic algorithms, yielding competitive results with respect to performing these selections separately.

3 Methods

3.1 AdaBoost

AdaBoost is an ensemble method that dynamically updates the sample weights using the training error of a given base classifier to train the next ensemble base classifier [29]. The process to construct an AdaBoost ensemble is described in Algorithm 1: For a set of N training samples x_i with labels y_i, a uniform distribution of weights w_i, is initialized. In step (1) a base classifier is trained on the weighted samples and in step (2) the training error is calculated. Then, the weight α_t for the base classifier $h_t(x)$ is computed as a function of the training error in step (3). The sample weights, subject to normalization, are updated in step (4). Samples which are correctly classified obtain lower weights and misclassified samples obtain higher weights; in this manner, samples harder to classify receive more attention

by the algorithm. More classifiers are added sequentially to the ensemble, each one correcting its predecessor. This process is repeated T cycles. The predicted labels by the ensemble are given by the sign of the weighted sum of the base classifiers label predictions.

Algorithm 1 AdaBoost algorithm [29].

1: **Input:** training samples and their labels $\{x, y\}$, base classifier $h(x)$ and number of cycles T.
2: **Initialize:** training weights, $w_i^1 = 1/n$, for $i = 1, ..., n$.
3: **for** $t = 1, ..., T$ **do**
4: (1) Weight the samples with weights w_i^t to train base classifier h_t.
5: (2) Calculate training error: $\epsilon_t = \sum_{i=1}^{n} w_i^t, y_i \neq h_t(x_i)$.
6: (3) Calculate score for h_t: $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$.
7: (4)
8: **for** $i = 1, ..., N$ **do**
9: update weights: $w_i^{t+1} = \dfrac{w_i^t e^{\{-\alpha_t y_i h_t(x_i)\}}}{Z_t}$,
10: where Z_t ensures the sum of weights to be unity.
11: **end for**
12: **end for**
13: **Output:** $H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$.

3.2 SVMs

In a binary SVM classifier, an optimal hyper-plane separating two classes in the feature space is found. During the optimization, the SVM model selects from the training samples, x_i, a subset of support vectors (SVs) that are used to establish the location of the decision surface. To simplify the search of the SVs, the training samples are mapped into a high-dimensional space using kernel functions, $k(x_i, x_j)$, which are expressed in terms of inner products of the training samples or their mappings. In the feature space, a specific kernel will yield a hyper-plane that is used to designate a prediction y_i to every x_i, depending on which side of the hyper-plane x_i is located. The kernel functions solve the optimization problem without explicitly utilizing the actual mappings; the latter is known as the kernel trick. Data may not be completely separable and some points would lie within the margin or they may be incorrectly classified; consequently, SVM implementations allow for a given degree of misclassification by introducing an adjustable penalty cost C.

An SVM classifier is defined by its kernel and the parameters that describe the kernel. This paper considers SVMs with:

- Radial Basis Function (RBF) kernel, $k(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2)$, with hyper-parameter γ.
- Sigmoid kernel, $k(x_i, x_j) = \tanh(\gamma\langle x_i, x_j\rangle + r)$ with hyper-parameters γ and r.
- Polynomial, $k(x_i, x_j) = (\gamma\langle x_i, x_j\rangle + r)^d$ with hyper-parameters γ, r and d.
- Linear kernel, $k(x_i, x_j) = \langle x_i, x_j\rangle$ with no hyper-parameters.

3.3 Boosted SVMs

3.3.1 AdaBoost ensembles

The main motivation to construct AdaBoost ensembles using SVMs as base classifiers resides in the fact that only few kernel hyper-parameters are needed to achieve a good ensemble performance; the latter simplifies the task of building an effective ensemble. X. Li et al. [21] studied AdaBoost SVM ensembles with RBF kernels. They showed that the hyper-parameter γ has the most relevant role in the ensemble classifier accuracy and they updated the sample weights by automatically modifying this parameter. Here, we constructed AdaBoost SVM ensembles with the kernels mentioned above and modified the kernel hyper-parameters and penalty cost. We set the different kernel hyper-parameters in such a way that the first base classifier in the ensemble achieves a classification accuracy slightly better than random choice. For the next base classifier, if the latter condition is not met, the hyper-parameters are adjusted to fulfill it. A moderate accuracy is desired, as an ensemble constructed in this fashion is expected to be more diverse and hence may have better generalization performance [21]. To determine how to adjust the kernel hyper-parameters and the penalty cost C in our AdaBoost sample weight updates, we studied the effect of different values and intervals of the hyper-parameters on a single SVM. In Fig. 1, we show the hyper-parameter dependency for the RBF, sigmoid, polynomial, and linear kernels trained with the `Abalone` data sample, described in Table 2. The plots show the training error as a function of the hyper-parameter γ and the cost penalty C. Each cell/number corresponds to a single SVM. The same behaviour is observed for SVMs trained with the remaining data sets listed in Table 2. From these studies, we concluded that γ and C should start with their minimum values of their ranges; subsequently, to improve the classification accuracy of the next classifier in the ensemble, we should increase γ and C towards their maximum values. The study suggests ranges and directions of change of the hyper-parameters as follows:

- For the RBF kernel, the ranges $\gamma \in (0, 100]$ and $C \in [0, 100]$ are used to cover a wide range.
- For the sigmoid kernel, the ranges $\gamma \in (0, 0.1]$, $C \in [0, 100]$ are used and we fix $r = -1$. The γ and r values

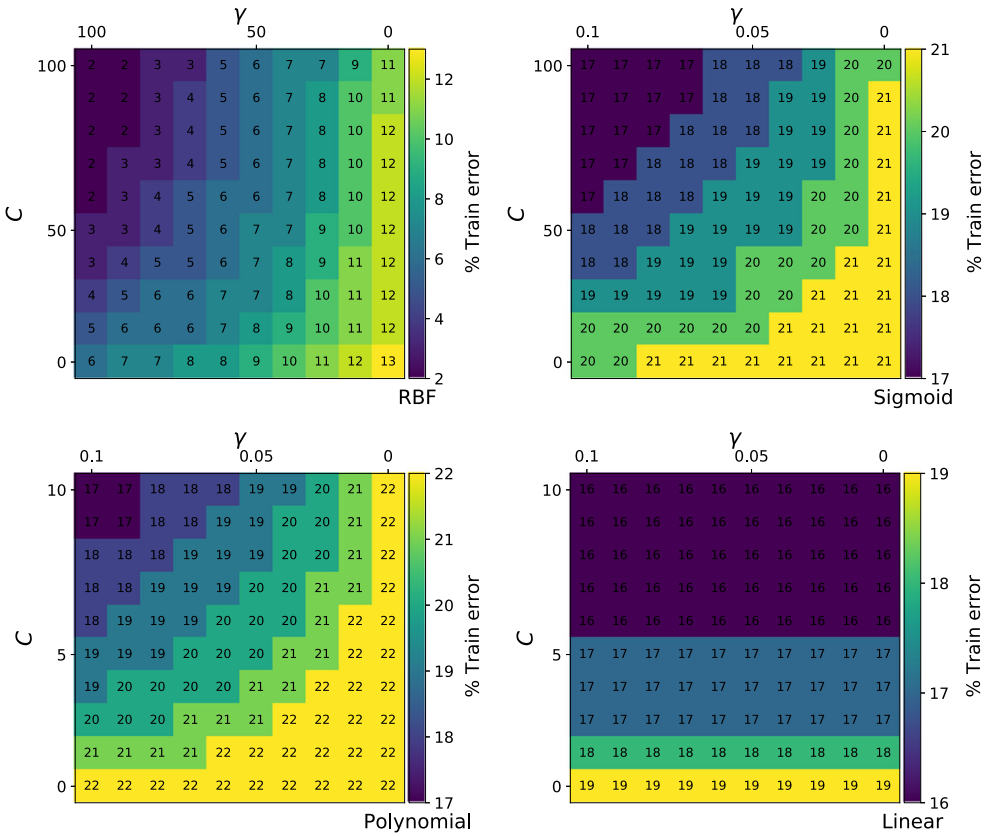

Fig. 1 SVM hyper-parameter study for the `abalone` data set. The plots show the change of the training error in percentage in function of γ and C for the RBF, sigmoid, polynomial and linear kernels, as indicated in each plot. The remaining hyper-parameters are fixed. The plots corresponding to the other data samples are not shown here, nevertheless they exhibit an analog behaviour

are chosen following Lin et al. [30]. The C values are chosen to cover a wide range.

- For the polynomial kernel, the ranges $\gamma \in (0, 0.1]$ and $C \in [0, 10]$ are used and we fix $r = 1$ and $d = 2$. The γ, r and d values are chosen following Chang et al. [31]. The C values are chosen to cover a wide range.
- For the linear kernel, the range $C \in [0, 10]$ is used to cover a wide range.

3.3.2 Ensemble diversity

Diversity in our AdaBoost process was carried out with the measure of diversity developed by Melville and Mooney [32], where the disagreement of an ensemble base classifier with the ensemble's prediction is taken into account for the measure. This diversity has proven to improve boosted ensembles with SVMs [21, 33]. Here, the ensemble diversity was implemented as follows: let $h_t(x_i)$ be the predicted label of the t-th base classifier upon the sample x_i, and the label predicted by all the existing base classifiers in the ensemble is given as $H(x_i)$. The diversity of the t-th base classifier is determined as

$$d_t(x_i) = \begin{cases} 0: & \text{if } h_t(x_i) = H(x_i) \\ 1: & \text{otherwise} \end{cases} \tag{1}$$

Expressly, we examine the disagreement in predictions between a new base classifier and the ensemble composed by the already incorporated base classifiers. The diversity of the whole ensemble with n samples and m components is calculated as

$$d_{ens} = \frac{1}{mn} \sum_{t=1}^{m} \sum_{i=1}^{n} d_t(x_i). \tag{2}$$

At each iteration, the ensemble diversity d_{ens} is calculated. If it is greater than a fixed threshold, the new base classifier is added to the ensemble, otherwise it is discarded and the cycle continues.

3.3.3 Ensemble regularization

To control the number of base classifiers in the ensemble, we considered a simple early stopping approach [34]. The stopping criteria rely on spotting the ensemble over-fitting that occurs within the training phase. The presence of an excessive number of classifiers in the ensemble makes the prediction accuracy on samples not used for training (test samples) decrease, diminishing the ensemble generalization. In this study, we state that over-fitting is reached when the test accuracy of an ensemble, with a given amount of base classifiers, decreases with respect

to the previous ensemble with a lower amount of base classifiers. To ensure that the over-fitting corresponds to a global ensemble behaviour and not a local inflection point, we stop the AdaBoost algorithm when N_{dr} consecutive generalization drops take place. Furthermore, to avoid the inclusion of more classifiers than needed, we stop the AdaBoost construction after N_{st} consecutive times the ensemble generalization exhibits the same value. Finally, we set T_{max} as the maximum amount of base classifiers allowed in the ensemble.

3.4 Genetic algorithms and SVMs

3.4.1 Genetic selection with support vectors

The GAs are stochastic optimization techniques inspired by the principles of the biological theory of evolution. GAs carry out selections with a binary representation and simple operators based on genetic recombinations and mutations [35]. In this work, we employ a GA to select a small subset of the training data that will likely contain the support vectors needed to address the binary classification problem. First, from the total training data, a group of N_c chromosomes[1] referred to as population, is initialized. Later, a fitness function, SVM, determines how good a chromosome is in classifying new data and assigns a fitness score considering a given performance metric. Once all the chromosomes are evaluated, the population is ranked by fitness score. Next, the selection process takes place. Two chromosomes, or parents $P_{a,b}$, are selected according to a specific genetic selection method. The two parents are used to generate a new offspring of length N_c. The new set of chromosomes is constructed by randomly selecting genes from both P_a and P_b. This stage of the GA is known as crossover. Afterwards, a mutation procedure is performed by replacing random vectors from the new offspring with vectors from the total data set, with a mutation rate R_m. Then, the new generation is appended to the original population and ranked by fitness. Only the fittest N_c individuals are kept, preserving the original population size. These steps are repeated until a termination criterion is reached to ensure that the scores are the best possible scores achieved by the algorithm; that is, it does not produce a new offspring which is significantly different from the previous generation. During the whole process, class balance is preserved.

[1] In this context a chromosome, made of l_c genes, consists of a set of randomly selected vectors which represent two different classes, thus a gene is a vector or data point. The chromosomes are restricted to contain an equal number of each class.

3.4.2 Chromosome selection methods

High-low. Originally proposed in [36]. After the population is ranked by fitness, it is divided in two parts. Individual P_a is randomly selected from the high fit part, while P_b is randomly selected from the low fit part. The division point here is set to divide the population in two parts of equal length. Choosing a parent from the low fit part of the population promotes several different combinations and prevents early convergence.

Roulette wheel. This technique is analogue to a spinning roulette wheel, where all the chromosomes in the population occupy a portion of the roulette wheel according to their fitness values [37]. The fittest individual has the largest share of the wheel, while the weakest individual has the smallest share of the roulette wheel. Two parents, P_a and P_b, are selected with this method by spinning the wheel.

Tournament. Individuals are chosen at random from the total population and set to compete, the winner being the one with the highest fitness score [37]. First, a number k of competitors is chosen and the individuals are randomly drawn without replacement. Then the fittest competitor is selected as parent P_a and withdrawn from the total population. Parent P_b is selected similarly.

3.5 GenBoost-SVM

Our proposed method, GenBoost-SVM, implements the elements described within the previous sections. Algorithm 2 GenBoost-SVM proceeds as follows:

- *Global Input*. Input the original sample vectors and labels $\{x, y\}$ and define the SVM kernel with its hyper-parameters as in Section 3.3.1. Include the ensemble diversity and regularization as in Section 3.3.2 and Section 3.3.3, respectively. The input also requires the genetic selection criteria, which defines the chromosome length l_c, the population size N_c, the chromosome ranking metric, the mutation rate R_m as in Section 3.4.1, and the selection method for parents P_a and P_b as in Section 3.4.2.
- *Genetic selection*. Create a population from $\{x, y\}$ comprising N_c chromosomes of length l_c. Start a cycle until the termination criterion of Section 3.4.1 is met: In step (1), define an AdaBoost ensemble with Algorithm 1 and weights updated as the SVM kernel hyper-parameters vary according to the input ranges. Provide the ensemble diversity and regularization criteria. In step (2), use a given classifier metric to asses the performance of the ensemble for each chromosome in the population. Rank the chromosomes by score from

low to high and keep N_c of them. In step (3), select parents $P_{a,b}$ with the input selection method. In step (4), crossover the parents $P_{a,b}$ to generate a new offspring and then mutate them with a mutation rate R_m. Append the new chromosomes to the population. After the cycle is terminated, a small subset $\{x', y'\}$ is obtained.

- *Final training.* With the obtained $\{x', y'\}$, train an ensemble of SVM classifiers once, using the Algorithm 1 and same kernel, hyper-parameters, ensemble diversity and regularization used for the genetic selection.
- *Global output:* The output is a final classifier trained with a reduced subset. Its prediction is given by the sign of the linear combination of SVMs with AdaBoost weights. This ensemble classifier is to be tested with the vectors not included in $\{x', y'\}$.

Algorithm 2 GenBoost-SVM.

1: **Global input:** Sample vectors and labels $\{x, y\}$, SVM classifier, ensemble diversity and regularization criteria, genetic selection parameters.
2: **Genetic selection:**
3: - Input: chromosome population created from $\{x, y\}$.
4: **while** *genetic selection termination criterion* **do**
5: (1) Determine chromosome fitness with Algorithm 1. Use the SVM as base classifiers and include ensemble diversity and regularization.
6: (2) Assign scores and sort the population by score.
7: (3) Select parents P_a and P_b.
8: (4) Generate offspring and mutation.
9: **end while**
10: - Output: Reduced set of training vectors and labels $\{x', y'\}$.
11: **Final training:**
12: - Input: Training vectors and labels $\{x', y'\}$.
13: - Train an ensemble with Algorithm 1. Use the SVM as base classifiers and include ensemble diversity and regularization criteria.
14: **Global output:** $H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$.

4 Experimental setup

4.1 Classifier metrics

4.1.1 Metrics

The present paper is restricted to the simple case where the data is divided in two different classes; that is, binary classification. We follow the convention of labelling these classes as ± 1. It is possible to summarize the outcome of a classifier using an error matrix [38]. The error matrix elements are defined in function of the classifier's ability to correctly (incorrectly) classify, as either positive or negative, a data point skipped during the training phase. In this paper, the error matrix elements are:

- TP is the number the positive samples correctly classified (true positives);
- FP is the number the negative samples incorrectly classified as positive (false positives);
- FN is the number the positive samples incorrectly classified as negative (false negatives);
- TN is the number the negative samples correctly classified (true negatives).

The following classifier metrics, in terms of the error matrix elements, are used through this document:

- Accuracy ACC [39]. The ACC is the number of correct predictions divided by the total number of predictions.
- Precision-recovery PRC [40]. The PRC plot shows the precision values (positive predictive value) for the corresponding sensitivity values, is given by PRC $= TP/(TP + FP)$.
- Area under the receiver operating characteristic curve AUC [41]. This curve is a plot of the TP rates at different thresholds. The thresholds for the SVM are obtained when the offset of the hyper-plane from origin is varied to produce different predictions. In the ensembles, we vary the thresholds of the component base classifiers. The area under the curve is calculated with a trapezoidal integration.

The values of the metrics ACC, PRC and AUC are in the range [0, 1] where 0 corresponds to the worst performance and 1 to the best performance.

4.1.2 Cross-validation

To provide reliable performance metrics for each ensemble, we carried out a repeated k-fold cross-validation procedure [42]. We split the data samples into k different folds. The k^i fold is used to test the model and yield the performance metrics, whilst the remaining folds are used for training the model. This process is done for each of the k-folds. The k-fold cross-validation is repeated N_{cv} times with a different k^i random generation in every repetition. In total, we train and test our models $k \times N_{cv}$ times. The reported metric values are the mean values of the obtained distributions and we assign one standard deviation as their errors.

4.2 Model construction

4.2.1 Model combinations

We study a number of ensembles constructed using the elements of Section 3. First, we set whether to use a given genetic pre-selection or the traditional test-train data splitting. The ensembles that include genetic selections need a metric to assort the fittest vectors in the population and select the parents $P_{a,b}$. We studied the two fitness metrics ACC and AUC as the assorting metrics. Next, the ensemble diversity is defined exploring two possibilities: when there is no requirement for the ensemble diversity or when we require a threshold in Eq. 2 to add a new classifier in the ensemble. Finally, we select one of the four SVM base classifier's kernels described in Section 3.3.1. Therefore, for each of the three genetic selection methods of Section 3.4.1, there will be 16 different manners of how to construct an ensemble. That is, $2 \otimes 2 \otimes 4$=(AUC, ACC)$\otimes$(Diversity, No-Diversity)$\otimes$(RBF, sigmoid, polynomial, linear), where the first term of the right side of this equality corresponds to the metrics used for the genetic selection method, the second term to the ensemble diversity and the third term to the SVM kernel considered. Additionally, 8 combinations are included in our studies which do not use a genetic selection and are described with a similar equality, $2 \otimes 4$=(Diversity, No-Diversity)$\otimes$(RBF, sigmoid, polynomial, linear). We refer to these selections as traditional selections. Overall, we study $3 \otimes 16 \oplus 8 = 56$ different combinations. The ensembles adopt a naming scheme as follows: `selection-kernel-diversity`. In Appendix A we give a glossary of the abbreviations used. For example, an ensemble that uses a genetic selection with the high-low method is named as `genHLAUC-RBF-YESdiv`. This ensemble considers the AUC as the fitness metric and the RBF as the SVM kernel and fulfills the diversity criteria. There is an equivalence with the ensembles proposed by X. Li et al. [21]. That is, our ensembles `trad-rbf-NOTdiv` and `trad-rbf-YESdiv` correspond to the *AdaBoostSVM* and *Diverse AdaBoostSVM* algorithms respectively proposed in their work.

4.2.2 Ensemble parameters

To fully define our proposed ensembles, it is necessary to fix the ensemble parameters. A summary of these parameters is shown in Table 1 and are explained in the following:

- *AdaBoost ensemble.* We state that the base classifier is an SVM. Next, we set the threshold diversity as described in Section 3.3.2, $d_{ens} > 0.7$ which ensures a highly diverse ensemble. The maximum number of base classifiers T_{max} is 250, which avoids an excessive number of classifiers that may cause over-training and long training times. The number of performance drops, $N_{dr} = 4$, ensures that the reduction of the ensemble performance is a real effect. We stop adding base classifiers to the ensemble when the performance metrics stall after $N_{st} = 10$ times. This quantity will ensure that no ensemble improvement is possible.
- *SVM.* For the four different types of SVM kernels, we set C and γ as described in Section 3.3.1. The remaining hyper-parameters are kept fixed.
- *Genetic selection.* The population size is set to $N_c = 10$, which will give enough alternatives to select the parents $P_{a,b}$. The chromosome length l_c is given in terms of the total number of data vectors n. By letting $l_c = n/4$, we can achieve reasonable statistics for low statistics samples and reduce the training time for high statistics samples. We use AUC and ACC as two different chromosome ranking metrics in order to compare the effect of choosing a specific ranking metric. Different parent selections are considered in order to compare the effect of the popular parent selections, as described in Section 3.4.1. The mutation rate $R_m = 0.25$ means that one quarter of a given chromosome will mutate, whilst the rest will stay the same. This number is reasonable since the fraction of not-mutated genes will preserve the essence of the selected parents while including an effective mutation. The number of repetitions of the GA selection is driven by the stop criterion of Section 3.4.1.
- *Cross-validation.* The 10 k-folds and the repetitions $N_{cv} = 5$ are chosen to produce enough statistics and provide reliable standard deviations of the performance metrics, while keeping the number of training times computationally reachable.

4.3 Data samples

The data samples used to test our models are taken from the UCI repository [43]. Their characteristics are summarized in Table 2. For the case when they exhibit categorical features, we convert these categories to numerical values by assigning integers in ascending order to every category. Afterwards, the features of the data samples are transformed to lie in the range [0,1], where the SVM classifiers perform the best [44]. Some data sets contain more than two classes, in which case we grouped them to allow only two classes. The last column of Table 2 contains the imbalance measures of the data samples. In a data sample labeled with two different classes, we define the imbalance as the ratio of the number of examples with lower presence to the number of examples with higher presence. Therefore, the imbalance measure is

Table 1 Parameters used to construct and characterize our proposed ensembles

	Parameter	Value
AdaBoost	Base classifier	SVM
	Diversity threshold for d_{ens}	0.7
	Max base classifiers T_{max}	300
	Training metric drops N_{dr}	4
	Training metric stalls N_{st}	10
SVM	Cost C	As described in Section 3.3.1
	Gamma γ	As described in Section 3.3.1
	Remaining hyper-parameters	Fixed
	Kernel	RBF, polynomial, sigmoid, linear
Genetic selection	Population size N_c	10
	Chromosome length l_c	$n/4$
	Fitness evaluation	AdaBoost-SVM
	Ranking metrics	AUC, ACC
	Parent selection	As in Section 3.4.2
	Mutation rate R_m	0.25
	Number of generations	Stop as described in Section 3.4.1
Cross-validation	k-folds	10
	Repeteitions N_{cv}	5

in the range [0,1], where a value close to 1 corresponds to a fully balanced data sample and a value close to 0 corresponds to a highly imbalanced data sample.

4.4 Experimental implementation

The experiments were carried out within `Python` using `NumPy` [45] and the `libsvm` [44] implementation from `scikit-learn` [46]. The statistical tests of the repeated cross-validation results are performed using the `SciPy` [47] tools. Our experiments follow Algorithm 2, with the ensemble construction guided by Section 4.2.1, with the parameters of Table 1. Our code is publicly available on `GitHub` [48].

Table 2 UCI [43] data sets used in the experiments

Sample	+1 Class	−1 Class	N features	Imbalance
`titanic`	342	542	7	0.63
`cancer`	201	85	9	0.42
`german`	800	300	20	0.38
`heart`	160	137	13	0.86
`solar`	865	201	10	0.23
`car`	518	201	6	0.39
`ecoli`	143	193	7	0.74
`wine`	4715	183	11	0.04
`abalone`	960	3217	8	0.30
`adult`	7508	22654	15	0.33

5 Results

5.1 Ensemble metrics

In Fig. 2 we present the ACC, PRC, AUC, total training time, number of classifiers, and number of training vectors produced by our 56 proposed ensembles across the data samples listed in Table 2. Each point in these plots represents the mean value of the $k \times N_{cv}$ cross validation calculated metrics. The ACC, PRC, AUC are computed with the predicted classes from the testing samples which were not included while training the models. The number of classifiers corresponds to the number of base classifiers that each ensemble acquired during the execution of our algorithm. The number of training vectors is the number of input vectors $\{x, y\}$ for the ensembles which do not consider a genetic selection. For the ensembles that consider a genetic selection, the number of training vectors is the number of the reduced set of training vectors $\{x', y'\}$ when applying the genetic selection of Algorithm 2. The training time is the total time to complete Algorithm 2. The solid

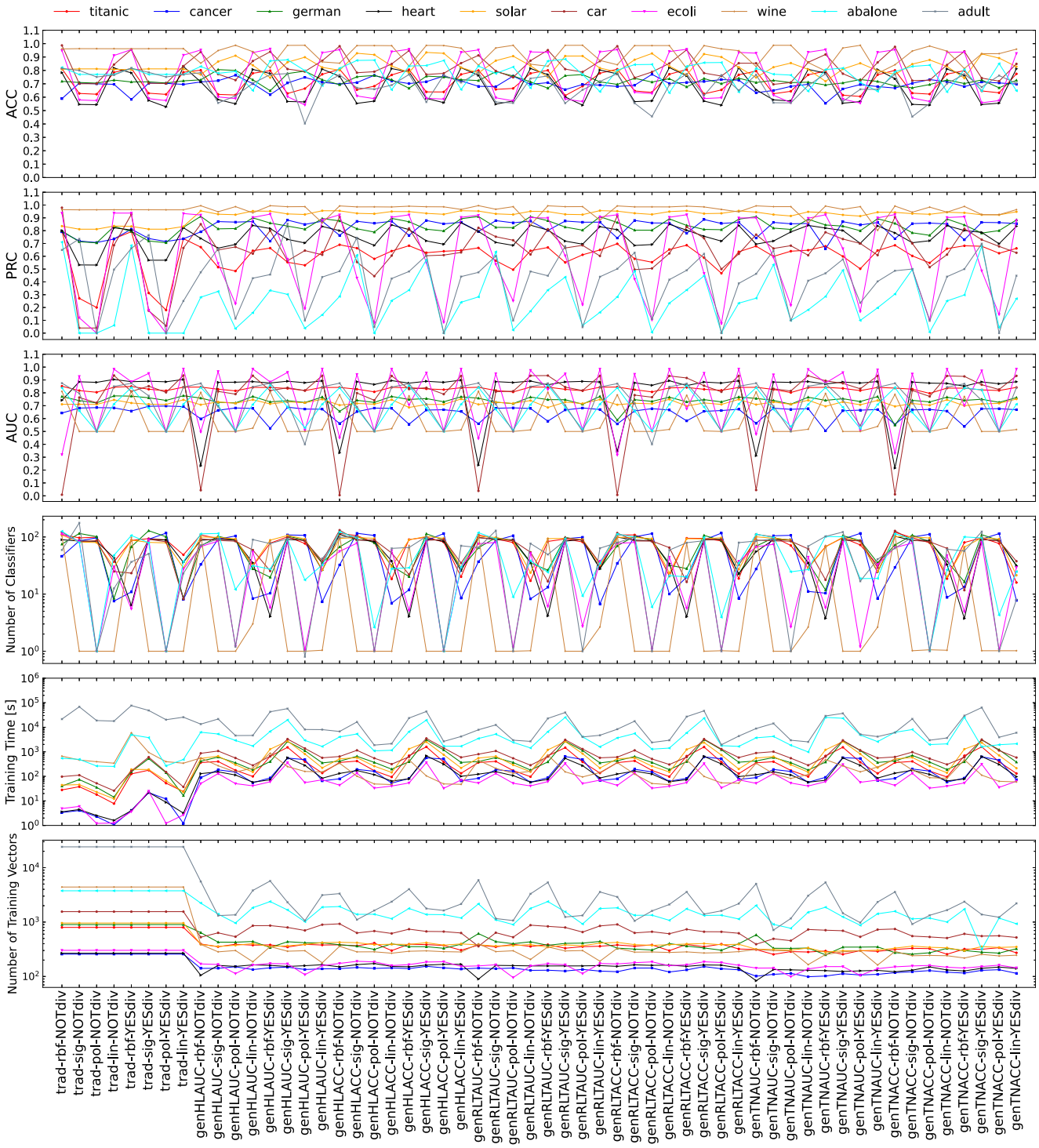

Fig. 2 ACC, PRC, ACC, the number of classifiers in the ensemble, the training time and training vectors for our proposed ensembles. The x-axis lists the 56 ensembles following the name scheme `selection-kernel-diversity`. The points are the mean values of the $k \times N_{cv}$ repeated cross validation computations. Each marker and color correspond to a data sample

lines rendered in the plots join the results of a given data sample in such a way that a solid line displays the general behaviour of our ensembles evaluating a specific data sample.

5.2 Best ensembles combinations

To provide insights on how well our proposed ensembles perform, we carefully studied a selection of them, as it is

impractical to handle the 56 combinations. Therefore, using the samples listed in Table 2, we empirically conducted a selection to obtain the best ensembles. Our procedure is as follows: First, we computed the ACC and sorted the values from high to low. Next, the AUC and PRC were calculated and we only considered ensembles where their ACC and PRC is higher than the average of the 56 ensemble combinations. Then we selected the classifiers with the highest ACC and repeated these steps for each sample in Table 2. We selected four ensembles which appeared the most often within the ensembles displaying the highest ACCs in each data sample:

- Ensemble `trad-rbf-YESdiv` (E1). Here, the traditional selection and data splitting procedure to divide the sample into training and testing is employed. The RBF kernel is used and diversity is considered. Recall that E1 is equivalent to the X. Li et al. [21] *Diverse AdaBoostSVM* ensemble.
- Ensemble `genHLACC-rbf-NOTdiv` (E2). A subset of training vectors is selected with the high-low genetic selection method. For this ensemble the fitness metric is the ACC. The RBF kernel is used and the ensemble diversity constrains are not included.
- Ensemble `genHLAUC-sig-YESdiv` (E3). A subset of training vectors is selected with the high-low genetic selection method. For this ensemble the fitness metric is the AUC. The RBF kernel is used and the ensemble diversity constrains are included.
- Ensemble `genHLACC-pol-YESdiv` (E4). A subset of training vectors is selected with the high-low genetic selection method. For this ensemble the fitness metric is the ACC. The polynomial kernel is used and the ensemble diversity constrains are included.

5.3 Performance comparison

We compare the ACC, PRC and AUC of our selected best ensembles against the PCs with the aid of a paired ranked Wilcoxon statistical test [49]. This test is the analogue of the Student t-test for data showing non-Gaussian shapes. The test will assert whether the difference of a given metric between two classifiers is statistically significant. The Wilcoxon test performs a hypothesis test where the null hypothesis, H_0, refers to the case when the classifiers metrics are equal. Namely, we reject H_0 setting by $\alpha = 0.05$, that is, we claim that the AUC, ACC and PRC of two classifiers are not equal if the p-value < 0.05.

5.3.1 Comparison with popular classifiers

The PCs we considered and their parameters are summarized in Appendix B. Tables 3, 4 and 5 show the Wilcoxon statistical tests for the `wine` sample (the tables corresponding to this exercise were produced as well for the data samples in Table 2 but not shown here). In these tables, the mean value μ and the standard deviation σ of the $k \times N_{cv}$ metrics calculated for our ensembles and the PCs are shown and compared. Here, a cross mark (✗) indicates that we have failed to reject the null hypothesis H_0. Conversely, the check mark (✓) indicates that it is possible to reject H_0 with the established confidence level. In this way, the blue check mark represents the case when the difference of the mean value of the metrics between one of our proposed classifier and a PC is positive. The red check mark indicates that this difference is negative. Concisely, the (✗) mark states that our selected ensemble and the PC are not different, the blue (✓) mark states that our selected ensemble outperforms the PC, and the red (✓) mark states that the PC outperforms our selected ensemble.

5.3.2 Summary tables

Tables 6–8 condense the Wilcoxon statistical tests. In these tables, we summarize the results of Tables 3–4 and the ones corresponding to the rest of the samples of Table 2. Tables 6–8 contain the number of times we are able to reject H_0 when comparing our selected ensembles E1, E2, E3, E4 (see Section 5.2) against all the considered PCs. Specifically, N^+_{rej} is the number of times we rejected H_0, when the differences of the mean values of the performance metrics between one of our selected ensembles and the PCs, are positive. Similarly, N^-_{rej} represents the number of times when these differences are negative. Therefore, each count in N^+_{rej} implies that our selected ensemble performs better with respect to a given PC. On the other hand, each count in N^-_{rej} implies that our selected ensemble performs worse than a given PC. Hence, when $N^+_{rej} \geq N^-_{rej}$, we could state that our selected ensemble is exhibiting a competitive or better performance compared with the considered PCs. Tables 6–7 correspond to classifiers metrics, ACC, AUC and PRC, respectively. The bottom row in these tables adds up the values of each data sample.

5.4 Discussion

5.4.1 Ensemble performance

In Fig. 2 it is found that most of the ACC, PRC and AUC values are above 0.5, meaning that our ensembles perform better than random guess. Another general feature is that the performance of the ensembles varies according to the data sample. This is expected since every data sample is pre-processed in the same manner to avoid biases arising from knowing too well a concrete data sample. In particular,

Table 3 Wilcoxon tests for the `wine` sample using the ACC metric

rbf-svm	0.96(0.01)	✓	✓	✓	✓
poly-svm	0.96(0.01)	✗	✓	✓	✓
sigmoid-svm	0.96(0.01)	✓	✓	✓	✓
linear-svm	0.96(0.01)	✓	✓	✓	✓
bag-svm	0.97(0.01)	✓	✓	✓	✓
rand-forest	0.97(0.01)	✓	✓	✓	✓
bdt-forest	0.96(0.01)	✗	✓	✓	✓
bag-forest	0.97(0.01)	✓	✓	✓	✓
grad-forest	0.96(0.01)	✗	✓	✓	✓
neural-net	0.96(0.01)	✗	✓	✓	✓
k-neigh	0.96(0.01)	✗	✓	✓	✓
gauss-nb	0.94(0.01)	✓	✓	✓	✓
gauss-pc	0.96(0.01)	✓	✓	✓	✓
log-reg	0.96(0.01)	✓	✓	✓	✓
ridge-cl	0.96(0.01)	✓	✓	✓	✓
sgdc-cl	0.96(0.01)	✓	✓	✓	✓
pass-agre	0.96(0.01)	✗	✓	✓	✓
linear-dis	0.96(0.01)	✗	✓	✓	✓
quad-dis	0.94(0.01)	✓	✓	✓	✓

The metric mean value and its error are presented in the form $\mu(\sigma)$. The rejection of H_0 (R.H_0) is indicated with a (✓) mark (blue color when our ensemble outperforms the PC and viceversa for red color) and the failure to reject H_0 is indicated with a (✗) mark. Our selected ensembles E1, E2, E3, and E4 are compared with the considered PCs.

the ACC plot shows for the `cancer` and `german` data samples that the metric value is above 0.5 and stable for all our proposed ensembles. The PRC for the `solar`, `wine`, `cancer` and `german` data samples displays a similar stability. Finally, the AUC for the `titanic` and `solar` data samples are similarly stable. This performance stability shown for some samples means that training our proposed ensembles with different combinations utilizing these data

Table 4 Same as Table 3 but for the `wine` data sample and the PRC metric

rbf-svm	0.97(0.01)	✓	✓	✓	✓
poly-svm	0.96(0.01)	✗	✓	✓	✓
sigmoid-svm	0.96(0.01)	✓	✓	✓	✓
linear-svm	0.96(0.01)	✓	✓	✓	✓
bag-svm	0.97(0.01)	✗	✓	✓	✓
rand-forest	0.97(0.01)	✓	✓	✓	✓
bdt-forest	0.97(0.01)	✓	✓	✓	✓
bag-forest	0.97(0.01)	✓	✓	✓	✓
grad-forest	0.97(0.01)	✗	✓	✓	✓
neural-net	0.96(0.01)	✗	✓	✓	✓
k-neigh	0.97(0.01)	✗	✓	✓	✓
gauss-nb	0.97(0.01)	✓	✓	✓	✓
gauss-pc	0.96(0.01)	✓	✓	✓	✓
log-reg	0.96(0.01)	✓	✓	✓	✓
ridge-cl	0.96(0.01)	✓	✓	✓	✓
sgdc-cl	0.96(0.01)	✓	✓	✓	✓
pass-agre	0.96(0.01)	✗	✓	✓	✓
linear-dis	0.97(0.01)	✓	✓	✓	✓
quad-dis	0.97(0.01)	✓	✓	✓	✓

Table 5 Same as Table 3 but for the `wine` data sample and the AUC metric

rbf-svm	0.84(0.05)	✓	✓	✓	✓
poly-svm	0.79(0.06)	✓	✗	✓	✓
sigmoid-svm	0.48(0.07)	✓	✓	✗	✗
linear-svm	0.78(0.05)	✓	✗	✓	✓
bag-svm	0.85(0.04)	✓	✓	✓	✓
rand-forest	0.89(0.04)	✓	✓	✓	✓
bdt-forest	0.82(0.05)	✗	✓	✓	✓
bag-forest	0.89(0.03)	✓	✓	✓	✓
grad-forest	0.49(0.13)	✓	✓	✗	✗
neural-net	0.57(0.06)	✓	✓	✓	✓
k-neigh	0.71(0.04)	✓	✓	✓	✓
gauss-nb	0.8(0.04)	✗	✗	✓	✓
gauss-pc	0.8(0.05)	✓	✗	✓	✓
log-reg	0.78(0.05)	✓	✗	✓	✓
ridge-cl	0.78(0.05)	✓	✗	✓	✓
sgdc-cl	0.77(0.06)	✓	✓	✓	✓
pass-agre	0.77(0.05)	✓	✓	✓	✓
linear-dis	0.78(0.05)	✓	✗	✓	✓
quad-dis	0.79(0.05)	✓	✗	✓	✓

samples, does not significantly improve the ensembles' ability to describe the data.

A more interesting behavior is when the values of the ACC, PRC, and AUC drastically change across different ensembles. There are ensembles which fail to model specific data samples as they perform worse than random guess. A clear example of the changing pattern arises for the PRC for `ecoli`, `abalone` and `adult` data samples, since some ensembles achieve metrics well above 0.5. This indicates that these ensembles are capable to model correctly these data samples, whilst other ensembles yield metrics close to 0 failing to describe the data samples. For these samples, we note that the PRC yielded by ensembles which use the SVM polynomial kernel (`pol`) is near zero regardless of the genetic selection or the inclusion of diversity criteria. Ensembles that use any other SVM kernel have an increased PRC. These changes of performance, along with the determination that the ensemble `genHLACC-pol-YESdiv` as one of our best ensembles (see Section 5.2), indicates that for some samples, the use of the SVM polynomial kernel should be avoided, whilst for other samples it is the most appropriate choice. Another interesting feature observed in Fig. 2, is that the AUC value reduces to zero for the ensembles that use the SVM RBF kernel and the diversity criteria is not included (`rbf-NOTdiv`) in the

Table 6 Summary table for ACC and all data sets of Table 2

ACC	E1		E2		E3		E4	
Sample	N^+_{rej}	N^-_{rej}	N^+_{rej}	N^-_{rej}	N^+_{rej}	N^-_{rej}	N^+_{rej}	N^-_{rej}
`titanic`	7	1	6	1	1	18	1	18
`cancer`	0	18	19	0	2	0	19	0
`german`	3	10	0	14	19	0	19	0
`heart`	3	11	3	14	0	16	0	16
`solar`	3	7	3	2	19	0	19	0
`car`	15	2	17	2	3	15	3	16
`ecoli`	6	0	6	6	2	16	0	16
`wine`	2	10	0	19	19	0	19	0
`abalone`	10	5	6	0	19	0	19	0
`adult`	6	12	2	17	2	17	2	17
Total	55	76	62	75	86	82	101	83

Table 7 Summary table for PRC and all data sets of Table 2

PRC	E1		E2		E3		E4	
Sample	N^+_{rej}	N^-_{rej}	N^+_{rej}	N^-_{rej}	N^+_{rej}	N^-_{rej}	N^+_{rej}	N^-_{rej}
`titanic`	10	0	1	18	1	18	1	3
`cancer`	16	0	19	0	19	0	19	0
`german`	5	1	19	0	16	0	4	3
`heart`	3	0	3	6	3	16	3	16
`solar`	5	6	19	0	19	0	19	0
`car`	11	3	11	4	3	13	3	11
`ecoli`	4	0	4	15	1	15	1	18
`wine`	6	7	19	0	19	0	19	0
`abalone`	6	8	1	18	0	17	0	19
`adult`	3	16	2	17	19	0	0	19
Total	69	41	98	78	100	79	69	89

`ecoli`, `car` and `heart` samples. This issue is fixed when diversity is included, demonstrating its importance in some ensembles (`rbf-YESdiv`), even when in some samples the diversity effect may appear negligible. This is in agreement with our selected classifiers in Section 5.2, where three out four of the selected ensembles are constrained by the diversity criteria.

The number of classifiers plotted in Fig. 2 shows that some ensembles contain up-to 10^2 base classifiers, whilst other ensembles contain only contain a few of them. For the `ecoli`, `abalone`, and `adult` data samples the ensembles constructed with the SVM polynomial kernel contain only a few base classifiers. The latter should be related with the poor performance above mentioned for these samples, in which case the ensemble construction has failed. On the other hand, note that for the ensembles which include the diversity criteria, the number of base classifiers is around 10^2 and the performance for ensembles of the type `rbf-NOTdiv` is increased, meaning that constructing ensembles including diversity is advantageous. Another pattern seen in the number of classifiers plot for the `titanic`, `solar`, `german`, and `heart`, is that the number of base classifiers changes as a function of the SVM kernel used: the number of base classifiers increases for the RBF, slightly increases for the sigmoid kernel, and finally it diminishes for the linear kernel. This has impact on the ACC since for the `solar` and `german` samples, the more classifiers in the ensembles, the higher the ACC values are, and for the `titanic` and `heart` the opposite behaviour is found. We conclude that the number of classifiers conforming the ensembles highly depends on the SVM kernel and that our early stop criteria have proven to be profitable.

Furthermore, in Fig. 2, the training time and number of training vectors plots show interesting features. First, the number of training vectors is stable for most of the samples (except for the `abalone` and `adult` samples). Naively, one may think that the training time should also display this stability. Nevertheless, the training time is more closely related to the number of classifiers acquired by the ensemble, that is, the more base classifiers in the ensemble, the longer the training times. The relation of the training time with the ACC, PRC and AUC is more complex: As mentioned above some samples are stable for a specific metric across all the constructed ensembles. Contrastingly, it is observed that for samples where the ACC, PRC and AUC metrics values change, the relation of these metrics with the training time should follow the above mentioned performance pattern dictated by the number of classifiers. For the `adult` data sample, several ensembles constructed using genetic selections see a reduction of the total training time. It is important to mention that the training times displayed in the plot, are the total times to complete Algorithm 2, where the genetic selection to find $\{x', y'\}$ stage is the lengthiest one. If the model training starts with $\{x', y'\}$ (already pre-selected training vectors), the training time for the ensembles that use this genetic selection is reduced to a few seconds as the number of training vectors is below 10^3 for most of the samples.

5.4.2 Performance comparison with popular classifiers

The results in Tables 6–8 provide information of how competitive are our ensembles compared with out-of-the-box PCs of Appendix B. The following contains comments regarding the $N^{+,-}_{rej}$ found for each studied data sample:

- `titanic`. Our selected ensembles make marginal improvements.
- `cancer`. E2 and E4 perform better than all the PCs for the ACC metric. For PRC our selected ensembles outperform the PCs.

Table 8 Summary table for AUC and all data sets of Table 2

AUC	E1		E2		E3		E4	
Sample	N^{+}_{rej}	N^{-}_{rej}	N^{+}_{rej}	N^{-}_{rej}	N^{+}_{rej}	N^{-}_{rej}	N^{+}_{rej}	N^{-}_{rej}
`titanic`	4	3	3	8	2	13	2	15
`cancer`	3	8	3	12	7	0	4	9
`german`	6	2	3	14	4	14	3	14
`heart`	5	0	0	19	5	4	4	8
`solar`	9	9	5	9	9	9	7	9
`car`	10	8	0	17	2	17	2	17
`ecoli`	1	14	0	19	2	14	0	19
`wine`	13	4	6	5	0	17	0	17
`abalone`	9	3	8	7	1	18	0	19
`adult`	5	11	6	11	2	17	1	18
Total	65	62	34	121	34	123	23	145

- `german`. E3 and E4 perform better than all the PCs for the ACC metric, while the opposite appears for E1 and E2. For PRC, E1 outperforms few PCs and on average the difference of our ensembles and the PCs is negligible; E2 and E3 exhibit the same or better performance than the PCs.
- `heart`. Our selected ensembles perform worse than most of the PCs for the ACC metric. A small number of PCs are outperformed by our ensembles for the PRC and AUC metrics.
- `solar`. E1 and E2 perform worse than the PCs for the ACC metric, whilst E3 and E4 mainly outperform the PCs. For PRC E2, E3, and E4 outperform all the PCs, and E1 shows negligible improvement.
- `car`. E1 and E2 perform better than most of the PCs for the ACC and PRC metrics, whilst E3 and E4 show the opposite behaviour. For AUC, only E1 outperforms a considerable amount of PCs.
- `ecoli`. E1 performs better against a few classifiers for the ACC metric, while showing similar performance for the remaining PCs. E2 outperforms some PCs, although is outperformed by the same number of PCs. E3 and E4 perform worse than most PCs. Only E1 shows a favorable number of outperformed PCs.
- `wine`. E1 and E2 perform worse than the PCs for the ACC metric, whilst E3 and E4 perform better than all PCs. For PRC, E2, E3 and E4 perform better than all the PCs.
- `abalone`. Our selected ensembles perform better than the PCs for the ACC metric. The PRC and AUC performance is worse in general than PC.
- `adult`. E3 outperforms all the PCs for the ACC metric. For the remaining ensembles and metrics, the PCs perform better than our ensembles.

Note that for the highly imbalanced `wine` and `solar` data samples, E3 and E4 fully outperform all the PCs for the ACC and PRC metrics. This enhanced performance may be achieved as our genetic selection effectively addresses the class imbalance challenge. Hence, our algorithm is appropriate to describe imbalanced data. Furthermore, for the high dimension `german` data sample, our algorithms yield a superior performance for some metrics when compared with the PCs. This may be achieved as the SVM are effective for treating high dimensional data. A comment about the AUC is that in our ensembles the AUC is worse than most of the PCs. The reason is that the definition of AUC in our ensembles is somewhat arbitrary and may need further studies to improve the treatment of our ensembles.

The global performance of our selected ensembles can be analyzed with the total $N^{+,-}_{rej}$, found at the bottom of Tables 6–8:

- In E1 ensemble `trad-rbf-YESdiv` the total counts for ACC is $N^{+,tot}_{rej} < N^{-,tot}_{rej}$; for PRC is $N^{+,tot}_{rej} > N^{-,tot}_{rej}$; and for AUC is $N^{+,tot}_{rej} > N^{-,tot}_{rej}$.
- In E2 ensemble `genHLACC-rbf-NOTdiv` the total counts for ACC is $N^{+,tot}_{rej} < N^{-,tot}_{rej}$; for PRC is $N^{+,tot}_{rej} > N^{-,tot}_{rej}$; and for AUC is $N^{+,tot}_{rej} < N^{-,tot}_{rej}$.
- In E3 ensemble `genHLAUC-sig-YESdiv` the total counts for ACC is $N^{+,tot}_{rej} > N^{-,tot}_{rej}$, for PRC is $N^{+,tot}_{rej} > N^{-,tot}_{rej}$; and for AUC is $N^{+,tot}_{rej} < N^{-,tot}_{rej}$.
- In E4 ensemble `genHLACC-pol-YESdiv` the total counts for ACC is $N^{+,tot}_{rej} > N^{-,tot}_{rej}$, for PRC is $N^{+,tot}_{rej} < N^{-,tot}_{rej}$; and for AUC is $N^{+,tot}_{rej} < N^{-,tot}_{rej}$.

In our selected ensembles there is a variety of SVM kernels. This is interesting since usually the RBF kernel is the most effective in the majority of cases. Thus,

switching between kernels may significantly enhance the description of a data sample without the need of a hyper-parameter search. Furthermore, in our ensembles the high-low genetic selection is preferred. This could be explained given the simplicity of the parents $P_{a,b}$ being selected with the high-low method to yield the reduced set $\{x', y'\}$. In our study, the roulette wheel and tournament selections show decent performances. Nevertheless, the high-low genetic selection is superior. Including ensemble diversity seems to be an important element that improves the final performance of our models. E1, E2, E3, and E4 perform better than the out-of-the-box PCs on average for the ACC and PRC metrics. Moreover, we compare their ratios $N_{rej}^{+,tot}/N_{rej}^{-,tot}$ and conclude that the best ensemble is E3 or `genHLAUC-sig-YESdiv`.

6 Conclusions

In this paper, we studied a methodology of how to construct ensembles using the AdaBoost technique with SVMs as base classifiers. We also explored the possibility of pre-selecting a smaller subset of the training data with genetic algorithms, which potentially contains the optimal vectors. In this way, the performance was improved and the training time of large samples was reduced in comparison with using a single support vector machine and finding the optimal hyper-parameters. We found that our ensemble `genHLAUC-sig-YESdiv` performs the best, suggesting that a genetic selection and dedicated ensemble construction help overcome the issues faced by single classifiers. Moreover, our best ensemble successfully describes imbalanced data. In future studies, we will explore the different variables that we kept fixed during this study. Additionally, the extension to multi-class classification might be explored for SVMs where the problem must be separated into several binary classifications. In this case, different hyper-surfaces are determined and hence an ensemble and a genetic selection are needed for each pair of classes. The latter poses the question of whether using the same kind of genetic selection and ensemble is better than using a mix of them in terms of performance and simplicity

Appendix A

Table A.1 Abbreviations of the elements used to construct the ensembles

Element	Description
`trad`	Traditional test-train data spliting
`genHL`	Data is pre-selected with the high-low genetic selection method
`genRLT`	Data is pre-selected with the roulette genetic selection method
`genTN`	Data is pre-selected with the tournament genetic selection method
`rbf`	The SVM base classifiers use an RBF kernel
`sig`	The SVM base classifiers use a sigmoid kernel
`pol`	The SVM base classifiers use a polynomial kernel
`lin`	The SVM base classifiers use a linear kernel
`YESdiv`	Ensemble diversity is included
`NOTdiv`	Ensemble diversity is not included

The names follow `selection-kernel-diversity` scheme

Appendix B

Table B.2 Popular classifiers used to compare with our proposed ensembles

Classifier	Description	Parameters
`rbf-svm`	Single SVM with RBF kernel	$\gamma = 1/N_{feat}, C = 1$
`poly-svm`	Single SVM with polynomial kernel	$\gamma = 1/N_{feat}, C = 1, d = 2, r = +1$
`sigmoid-svm`	Single SVM with sigmoid kernel	$\gamma = 1/N_{feat}, C = 1, d = 2, r = -1$
`linear-svm`	Single SVM with linear kernel	$C = 10$
`bag-svm`	SVM with RBF kernel bagging ensemble	$\gamma = 100, C = 1$
`rand-forest`	Decision trees random forest ensemble	$N_{DT} = 100$
`bdt-forest`	Decision trees AdaBoost ensemble	$N_{DT} = 100$
`bag-forest`	Decision trees bagging ensemble	$N_{DT} = 100$, sampling w/replacement
`grad-forest`	Decision trees gradient boosting ensemble	$N_{DT} = 100$, logistic regression classification
`neural-net`	Stochastic gradient neural network perceptron	$HL = 1, N_n = 100$, linear activation function
`k-neigh`	k-nearest neighbors classifier	$N_{neigh} = 3$

Table B.2 Popular classifiers used to compare with our proposed ensembles

Classifier	Description	Parameters
`gauss-nb`	Gaussian Naive Bayes	No priors considered
`gauss-pc`	Gaussian process with Laplace approximation	RBF kernel, L-BFGS-B optimization
`log-reg`	Logistic regression with L2 regularization	$C = 1$, L-BFGS-B optimization
`ridge-cl`	Ridge regression	$C = 1$
`sgdc-cl`	Linear stochastic gradient with L2 regularization	Linear SVM kernel
`pass-agre`	Passive aggressive	$C = 1$, PA-I optimization
`linear-dis`	Linear decision boundary	$NC = 1$, priors from data
`quad-dis`	Quadratic decision boundary	Gaussian density, priors from data

Acknowledgements This work was supported by the National Research Foundation of Korea (grants 2020R1I1A1A01066423, 2019R1I1A3A01058933, 2018R1A6A1A06024970).

Declarations

This work was supported by the National Research Foundation of Korea (grants 2020R1I1A1A01066423, 2019R1I1A3A01058933, 2018R1A6A1A06024970).

References

1. Zhang C, Ma YE (2012) Ensemble Machine Learning. Springer, https://doi.org/10.1007/978-1-4419-9326-7
2. Sagi O, Rokach L (2018) Ensemble learning: a survey. WIREs Data Min Knowl Discov 8(4):1249. https://doi.org/10.1002/widm.1249
3. Schapire RE (2003) The boosting approach to machine learning: an overview, pp 149–171. Springer, New York. https://doi.org/10.1007/978-0-387-21579-2_9
4. Wang W, Sun D (2021) The improved adaboost algorithms for imbalanced data classification. Inf Sci 563:358–374. https://doi.org/10.1016/j.ins.03.042
5. Wyner AJ, Olson M, Bleich J, Mease D (2017) Explaining the success of adaboost and random forests as interpolating classifiers. J Mach Learn Res 18:48–14833
6. Baig MM, Awais MM, El-Alfy E-SM (2017) Adaboost-based artificial neural network learning. Neurocomputing 248:120–126. https://doi.org/10.1016/j.neucom.2017.02.077. Neural Networks: Learning Algorithms and Classification Systems
7. Xu Y, Cong K, Zhu Q, He Y (2021) A novel adaboost ensemble model based on the reconstruction of local tangent space alignment and its application to multiple faults recognition. Journal of Process Control 104:158–167. https://doi.org/10.1016/j.jprocont.2021.07.004
8. Li W, Chen Y, Song Y (2020) Boosted k-nearest neighbor classifiers based on fuzzy granules. Knowl-Based Syst 195:105606. https://doi.org/10.1016/j.knosys.2020.105606
9. Liu Z, Liu D, Xiong J, Yuan X (2022) A parallel adaboost method for device-free indoor localization. IEEE Sens J 22(3):2409–2418. https://doi.org/10.1109/JSEN.2021.3133904
10. Taherkhani A, Cosma G, McGinnity TM (2020) Adaboost-cnn: an adaptive boosting algorithm for convolutional neural networks to classify multi-class imbalanced datasets using transfer learning. Neurocomputing 404:351–366. https://doi.org/10.1016/j.neucom.2020.03.064
11. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20:273–297
12. Schölkopf B, Smola AJ, Williamson RC, Bartlett PL (2000) New Support Vector Algorithms. Neural Computation 12(5):1207–1245. https://doi.org/10.1162/089976600300015565
13. Mavroforakis ME, Theodoridis S (2006) A geometric approach to support vector machine (svm) classification. IEEE trans neural netw 17(3):671–682
14. Awad M, Khanna R (2015) Support vector machines for classification. In: Efficient Learning Machines. Springer, pp 39–66, Berkeley, CA. https://doi.org/10.1007/978-1-4302-5990-9_3
15. Fan J, Zheng J, Wu L, Zhang F (2021) Estimation of daily maize transpiration using support vector machines, extreme gradient boosting, artificial and deep neural networks models. Agric Water Manag 245:106547. https://doi.org/10.1016/j.agwat.2020.106547
16. Hao P-Y, Kung C-F, Chang C-Y, Ou J-B (2021) Predicting stock price trends based on financial news articles and using a novel twin support vector machine with fuzzy hyperplane. Appl Soft Comput 98:106806. https://doi.org/10.1016/j.asoc.2020.106806
17. Viji C, Rajkumar N, Suganthi ST, Venkatachalam K, kumar TR, Pandiyan S (2021) An improved approach for automatic spine canal segmentation using probabilistic boosting tree (pbt) with fuzzy support vector machine. J Ambient Intell Humaniz Comput 12(6):6527–6536. https://doi.org/10.1007/s12652-020-02267-6
18. Al-Zoubi AM, Hassonah MA, Heidari AA, Faris H, Mafarja M, Aljarah I (2021) Evolutionary competitive swarm exploring optimal support vector machines and feature weighting. Soft Comput 25(4):3335–3352. https://doi.org/10.1007/s00500-020-05439-w
19. Zhou C, Chase JG, Rodgers GW (2021) Support vector machines for automated modelling of nonlinear structures using health monitoring results. Mech Syst Signal Process 149:107201. https://doi.org/10.1016/j.ymssp.2020.107201
20. Nalepa J, Kawulok M (2019) Selecting training sets for support vector machines: a review. Artif Intell Rev 52(2):857–900. https://doi.org/10.1007/s10462-017-9611-1
21. Li X, Wang L, Sung E (2008) Adaboost with svm-based component classifiers. Eng Appl Artif Intell 21(5):785–795. https://doi.org/10.1016/j.engappai.2007.07.001. Constraint Satisfaction Techniques for Planning and Scheduling Problems
22. Kim H-C, Pang S, Je H-M, Kim D, Bang S-Y (2002) Support vector machine ensemble with bagging. In: Lee S.-W., Verri A (eds) Pattern recognition with support vector machines. Springer, pp 397–408
23. Kawulok M, Nalepa J (2012) Support vector machines training data selection using a genetic algorithm. In: Gimel?farb G, Hancock E, Imiya A, Kuijper A, Kudo M, Omachi S, Windeatt T, Yamada K (eds) Structural, syntactic, statistical pattern recognition. Springer, pp 557–565

24. Nalepa J, Kawulok M (2014) Adaptive genetic algorithm to select training data for support vector machines. In: Esparcia-Alcázar AI, Mora AM (eds) Applications of evolutionary computation. Springer, pp 514–525
25. Nalepa J, Kawulok M (2014) A memetic algorithm to select training data for support vector machines. In: Proceedings of the 2014 annual conference on genetic and evolutionary computation. GECCO '14, pp 573–580, association for computing machinery, New York. https://doi.org/10.1145/2576768.2598370
26. Verbiest N, Derrac J, Cornelis C, García S, Herrera F (2016) Evolutionary wrapper approaches for training set selection as preprocessing mechanism for support vector machines: experimental evaluation and support vector analysis. Appl Soft Comput 38:10–22. https://doi.org/10.1016/j.asoc.2015.09.006
27. Fernandes ERQ, de Carvalho ACPLF, Coelho ALV (2015) An evolutionary sampling approach for classification with imbalanced data. In: International Joint Conference on Neural Networks (IJCNN). pp 1–7 https://doi.org/10.1109/IJCNN.2015.7280760
28. Kawulok M, Nalepa J, Dudzik W (2017) An alternating genetic algorithm for selecting svm model and training set. In: Carrasco-ochoa JA, Martínez-Trinidad JF, Olvera-López JA (eds) Recognition, Pattern. Springer, Cham, pp 94–104
29. Schapire RE, Singer Y (1999) Improved boosting algorithms using confidence-rated predictions. Mach Learn 37(3):297–336. https://doi.org/10.1023/A:1007614523901
30. Lin H-T, Lin C-J (2003) A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods. submitted Neural Comput 3(1-32):16
31. Chang Y-W, Hsieh C-J, Chang K-W, Ringgaard M, Lin C-J (2010) Training and testing low-degree polynomial data mappings via linear svm. J Mach Learn Res 11:1471–1490
32. Melville P, Mooney RJ (2005) Creating diversity in ensembles using artificial data. Inf Fusion 6(1):99–111. Diversity in Multiple Classifier Systems
33. Wu X, Lu X, Leung H (2019) A video based fire smoke detection using robust adaboost. Sensors 18:1–22
34. Prechelt L (1998) Early Stopping - But When?. Springer, pp 55–69. https://doi.org/10.1007/3-540-49430-8_3
35. Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor, MI. Second edn 1992
36. Elamin E (2006) A proposed genetic algorithm selection method. 1st National Symposium NITS
37. Goldberg DE (1989) Genetic Algorithms in Search, Optimization and Machine Learning, 1st edn. Addison-wesley Longman Publishing Co. Inc. USA
38. Powers D (2008) Evaluation: from precision, recall and f-factor to roc, informedness, markedness and correlation. Mach. Learn. Technol., vol 2
39. Foody GM (2002) Status of land cover classification accuracy assessment. Remote Sens Environ 80(1):185–201. https://doi.org/10.1016/S0034-4257(01)00295-4
40. Saito T, Rehmsmeier M (2015) The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. PloS one 10(3):0118432
41. Bradley AP (1997) The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern Recognit 30(7):1145–1159. https://doi.org/10.1016/S0031-3203(96)00142-2
42. Kuhn M, Johnson K (2013) Applied Predictive Modeling. SpringerLink: bücher, vol 26. Springer, New York
43. Dua D, Graff C (2017) UCI Machine Learning Repository http://archive.ics.uci.edu/ml
44. Chang C-C, Lin C-J (2011) LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2:27–12727. http://www.csie.ntu.edu.tw/cjlin/libsvm
45. Harris CR, Millman KJ, Van Der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, Van Kerkwijk MH, Brett M, Haldane A, del Río JF, Wiebe M, Peterson P, Gérard-Marchant P, Sheppard K, Reddy T, Weckesser W, Abbasi H, Gohlke C, Oliphant TE (2020) Array programming with numpy. Nature 585(7825):357–362. https://doi.org/10.1038/s41586-020-2649-2
46. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830
47. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, Polat İ, Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, Van Mulbregt P (2020) SciPy 1.0 contributors: sciPy 1.0: fundamental algorithms for scientific computing in python. Nature Methods 17:261–272. https://doi.org/10.1038/s41592-019-0686-2
48. Ramirez-Morales A, Salmon-Gamboa JU (2022) Genboost-svm code. https://github.com/andrex-naranjas/boosting
49. Wilcoxon F (1945) Individual comparisons by ranking methods. Biometrics Bulletin 1(6):80–83. https://doi.org/10.2307/3001968