31st International Conference on Artificial Neural Networks (ICANN), Bristol, UK, 2022

DOI: https://doi.org/10.1007/978-3-031-15937-4_25

SVM Ensembles on a Budget

David Nevado, Gonzalo Martínez-Muñoz[0000−0002−6125−6056], and Alberto Suárez[0000−0003−4534−0909]

Universidad Autónoma de Madrid, Escuela Politécnica Superior, Dpto. de Ingeniería Informática, C/Francisco Tomás y Valiente, 11, 28049 Madrid, Spain
{gonzalo.martinez,alberto.suarez}@uam.es

Abstract. This paper presents a model to train an ensemble of SVMs that achieves better generalization performance at a lower computational training cost than a single SVM. The idea of the proposed model is, instead of training a single SVM on the whole dataset, to train a diverse set of simpler SVMs. Specifically, the proposed algorithm creates B subensembles of T SVMs using a different set of hyper-parameters in each subensemble. Then, in order to gain more diversity, the T SVMs of each of the subsensembles are trained on a different $1/T$ disjoint fraction of the training set. The paper presents an extensive analysis of the computational training complexity of the algorithm. The experiments show that for any given computational budget, the presented method obtains a better generalization performance than a single SVM.

Keywords: SVM · ensembles of classifiers · computational complexity

1 Introduction

Support Vector Machines (SVM) are among the most accurate machine learning methods for classification problems [1, 5]. In spite of their excellent performance, they are costly to train. A cost that can be higher than quadratic with respect to the number of training instances [3]. Furthermore, the generalization capacity of an SVM is very sensitive to the actual values of its hyperparameters [4]. These are typically determined by a computationally expensive cross-validation search. For these reasons, the practical application of SVM's in large problems is limited. Ensembles of classifiers are a way to improve the performance of SVMs [12, 8, 7, 10]. Several strategies focus only in generating diverse SVMs for improving the accuracy of the model (for instance by using different kernels [12]). Other strategies are boosting-based optimization ensembles [8, 7]. Other works focus both in diversification and optimization [10]. However, the accuracy improvements are in general small as SVM are very stable classifiers [8].

In this work we present a novel architecture to build SVM ensembles with a low computational training complexity that can achieve similar or better accuracies than a single SVM trained on the complete dataset. In particular, given a limited training time budget, the proposed ensembles can have higher accuracy than a single SVM. In order to achieve this, a combination of two techniques is

used: Hyperparameter diversification and structured subsampling. These strategies serve to introduce the variability among the base SVMs that is needed to build an effective ensemble. Subsampling, in particular, not only fulfills this purpose of diversification, but also reduces the computational complexity of the training process. In addition, a detailed analysis of the computational cost of the proposed method is carried out.

2 Structured subbagging for SVM Ensembles

In order to achieve hyperparameter diversification, the proposed ensemble is composed of B subensembles of SVMs. The SVMs within each subensemble use an RBF kernel and are built using the same combination of hyperparameters (C, γ). However, each one of them is trained using a different random subset of size $tp \cdot N_{train}$, where tp is a hyperparameter of the ensemble that stands for *train proportion*. Hence, we need to generate B pairs of hyper-parameters $\{(C_b, \gamma_b)\}_{b=1}^{B}$, $\{(C_b, \gamma_b); b = 1, \ldots, B\}$, one for each subensemble. Each of these pairs (C_b, γ_b) is obtained by performing an exhaustive grid search with 2-fold cross-validation over an independent partition $\mathcal{D}_b$, sampled from the training set, $\mathcal{D}_{train}$. These partitions $\mathcal{D}_b$ have $2 \cdot N \cdot tp$ instances. In this way, during the grid search each SVM is trained on a sample of size $N \cdot tp$, the same number of instances on which each base SVM will be trained later. Having set the size of the partitions for searching for the hyper-parameters, $\mathcal{D}_b$, we distinguish two possible scenarios: If $tp \cdot B \leq 1/2$ then it is possible to extract from the training set $\mathcal{D}_{train}$, B disjoint subsets with $2 \cdot N \cdot tp$ instances each. If we have $tp \cdot B > 1/2$ then it is not possible to make such partitions. In this case we take B subsets of $2 \cdot N \cdot tp$ instances drawn from $\mathcal{D}_{train}$ uniformly at random without replacement.

Once the hyper-parameter pairs, $\{(C_b, \gamma_b)\}_{b=1}^{B}$, have been selected, B subensembles of SVMs are trained as follows: For each pair (C_b, γ_b) of hyper-parameters, the training set is randomly divided into $T = 1/tp$ disjoint subsets, $\{\mathcal{D}_t\}_{t=1}^{T}$, that is, $\mathcal{D}_{train} = \bigcup_{t=1}^{T} \mathcal{D}_t$. Then, one SVM with hyper-parameters (C_b, γ_b) is trained on each subset $\mathcal{D}_t$ for $t = 1, \ldots, T$. After iterating over the B hyper-parameter subsets and the $T = 1/tp$ subsets, a total of $B \cdot T$ SVMs will form the final ensemble. The complete pseudocode for this training procedure is detailed in Algorithm 1.

Note that in the ensemble method proposed in this work, the base models are trained on datasets that are extracted with a certain structure. Specifically, instead of using random sampling as in standard bagging or similar methods, the original dataset is partitioned into disjoint sets (see line 8 Alg. 1). By training each base learners on one of these disjoint subsets of the original training set, we expect to maximize the diversity of the base classifiers, and thus improve the benefits of aggregation. Furthermore, this method guarantees that all the samples available for training are used, which should limit the loss of information in the bootstap process and lead to an improved predictive performance. This is presented in the section on experiments. In order to validate these expected effects, an experiment is carried out to compare the differences of accuracy be-

Algorithm 1: SVM Ensemble training.

Input: $\mathcal{D}_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{train}}$ % Training set
B % Number of subensembles
tp % Train proportion
$grid$ % Hyperparameter grid

Output: E % Ensemble

if $tp \cdot B \leq 1/2$ **then**
  $\{\mathcal{D}_b\}_{b=1}^{B} \leftarrow$ Extract from $\mathcal{D}_{train}$ B disjoint subsets of size $2 \cdot tp \cdot N_{train}$.
else
  $\{\mathcal{D}_b\}_{b=1}^{B} \leftarrow$ Extract from $\mathcal{D}_{train}$ B subsets of size $2 \cdot tp \cdot N_{train}$ drawn uniformly at random.
$T = 1/tp$
foreach $b \leftarrow 1$ ***to*** B **do**
  $(C_b, \gamma_b) \leftarrow GridSearch(\mathcal{D}_b, folds = 2, grid)$
  $\{\mathcal{D}_t\}_{t=1}^{T} \leftarrow$ Split $\mathcal{D}_{train}$ into T disjoint subsets of size $tp \cdot N_{train}$
  foreach $t \leftarrow 1$ ***to*** T **do**
    % Train SVM with hyperparameters (C_b, γ_b) and data $\mathcal{D}_t$
    $c_{b,t} \leftarrow SVM(\mathcal{D}_t, (C_b, \gamma_b))$
$E(\cdot) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \sum_{b=1,t=1}^{B,T} \mathbb{1}[c_{b,t}(\cdot) = y]$
return E

tween ensembles built with the structured sampling strategy introduced in this work and standard subbagging.

The characteristics of ensemble learning algorithm depends on two hyperparameters B and tp, whose role we now review

- The ensemble consists in B subensembles. Each of these subensembles is composed of SVMs that are built using the same hyperparameters (C, γ). Therefore the SVMs within the same subensemble only differ in the data they are trained on. Exploratory experiments show that $B = 10$ is a good choice for most cases.
- The tp (*train proportion*) hyperparameter determines both the fraction of training samples that are used to build each base SVM of the ensemble and the number of SVMs to be combined in each subensemble. By design, each of the $T = 1/tp$ individual SVMs within a subensemble is built on a disjoint partition of the training set with $tp \cdot N_{train}$ instances. In this way, all training instances are used once for building each subensemble. In addition, since the datasets are disjoint, the diversity of SVMs should increase, an effect that is expected to improve the benefits of aggregation. Lower values of tp correspond to larger ensembles of SVMs trained on fewer instances. Higher values of tp correspond to smaller ensembles, with SVMs trained on larger samples. In the limit case, $tp = 1$, only one SVM is trained using the

whole dataset, which is equivalent to training a single SVM. The higher limit for possible values of T (low tp) is determined by the number of instances of the training set N_{train}. In this way, tp provides a way to balance between the strength of the individual classifiers and the size of the ensemble. Generally, $tp \in (0, 0.25]$ is a range of values that provide good performance.

Notice that the values of these hyperparameters jointly determine the size of the complete ensemble: With B subensembles each of size $1/tp$ we get a total ensemble size of B/tp.

3 Training complexity of the proposed model

The training complexity of the proposed model can be expressed as a function of four factors: N_{train}, $GSize$, B and tp, where N_{train} is number of samples of the training set, (B, tp) are the hyper-parameters of the model described in the previous section and $GSize$ is the size of the grid over which the hyperparameters (C, γ) are selected. In what follows, we assume that the cost of training an SVM on a dataset with N instances is $\approx \mathcal{O}(N^2)$, even though there is empirical evidence that it can be higher than quadratic [3]. Breaking down the training algorithm in its phases we get: Hyperparameters selection and base SVMs training.

The hyperparameters selection consists in an exhaustive 2-fold cross-validation search performed over the grid with folds of size $tp \cdot N_{train}$. This process is repeated B times. Ignoring the evaluation cost of the SVMs in the grid search, we get the following complexity for this phase

$$B \cdot 2 \cdot GSize \cdot \mathcal{O}(tp^2 \cdot N_{train}^2) \ . \tag{1}$$

In the base SVMs training phase, the ensemble is formed by B/tp SVMs trained on $tp \cdot N_{train}$ samples. The resulting cost is

$$B \cdot \frac{1}{tp} \cdot \mathcal{O}(tp^2 \cdot N_{train}^2) \ . \tag{2}$$

Adding up these costs we get an estimate of the overall complexity of the model

$$Cost = B \cdot \mathcal{O}(tp^2 \cdot N_{train}^2)(2 \cdot GSize + \frac{1}{tp}). \tag{3}$$

In most situations, we will have $2 \cdot GSize > 1/tp$. This implies that the search of hyperparameters phase is more demanding than training the base learners. The opposite can also happen for low values of tp, which may be adequate for very large datasets. This could also happen if the hyperparameter space is explored with alternative techniques such as Bayesian optimization [11], randomization or partial optimization [10] that lower the cost of exploring the hyperparameter grid. Considering that $T = 1/tp$, the final expression for the training complexity is

$$Cost = \mathcal{O}(B \cdot tp^2 \cdot N_{train}^2 \cdot (2GSize + T)) \ . \tag{4}$$

We can read this complexity as number of subensembles (B) times the training complexity of training a single SVM ($tp^2 \cdot N_{train}^2$) times the number of trained SVMs ($2GSize + T$) in each subensemble.

To put this results in perspective, we now detail the computational cost for training a single SVM on the whole training dataset for which a grid search is performed to tune its hyperparameters. We will assume an exhaustive grid search of K-fold cross-validation. As before, in this estimation we will ignore the evaluation cost of the SVMs during the grid search process.

The grid search involves the creation of ($GSize \cdot K$) SVMs, each of them trained on a fraction $(K-1)/K$ of the data. Hence, the total cost of this phase is:

$$GSize \cdot K \cdot \mathcal{O}\Big(\Big(\frac{K-1}{K}\Big)^2 N_{train}^2\Big), \tag{5}$$

and the cost of training the final SVM on the selected combination of hyperparameters is

$$\mathcal{O}\Big(N_{train}^2\Big). \tag{6}$$

In this case, it is clear that the hyperparameter tuning phase is the dominant term in the overall cost. Therefore we can write the estimation for the training cost of the single SVM as

$$SVM\_Cost = \mathcal{O}\Big(GSize \cdot \frac{(K-1)^2}{K} \cdot N_{train}^2\Big). \tag{7}$$

From the expressions (4) and (7) we observe that the training complexity for both models scale quadratically with the number of instances on the train set. If we suppose that $GSize > T$ (as is often the case), the ratio between the complexity of the proposed ensemble and a single SVM is

$$\mathcal{O}\Big(\frac{K \cdot B \cdot tp^2}{(K-1)^2}\Big). \tag{8}$$

This indicates that even though their training complexity have an equivalent asymptotic growth, the ensemble training cost can be adjusted using hyperparameter B and tp to achieve significant speedups. In section 4, we present the results of an empirical evaluation to validate the complexity estimations presented in this section.

4 Experimental evaluation

We now present the results of the experiments carried out to assess the generalization accuracy of the proposed ensemble as well as its complexity estimations. The comparison was carried out in six relatively large datasets for a single SVM: three synthetic datasets and three real datasets. The synthetic datasets [2] used are: *Twonorm*, *Threenorm* and *Ringnorm*. The real datasets (*Magic04*, *Bank Marketing* [9] and *Adult*) are taken form the UCI repository [6]. We did not considered larger datasets because training a single SVMs on datasets with more instances was unfeasible.

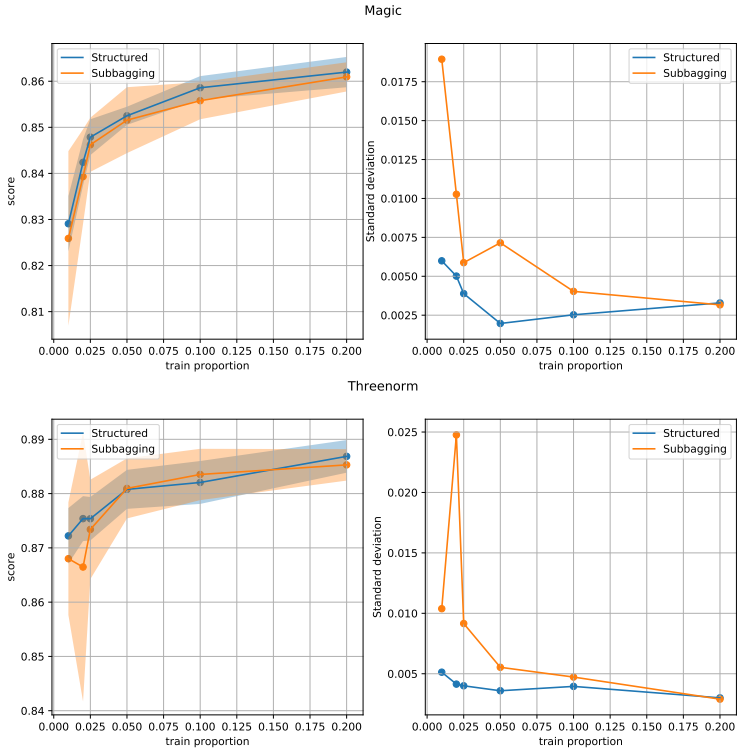

Fig. 1. Average accuracy (left column) and standard deviation (right) for structured sampling and subbagging models

Some basic preprocessing has been applied to the data as part of the training pipeline. Both Adult and Bank datasets have some categorical attributes. These attributes were transformed into numerical features using one-hot encoding. In the 6 problems all features were standardized to 0 mean and 1 standard deviation. All time measurements have been taken from single-threaded executions on an Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz with 8 GB RAM. All the hyperparameter searches for the SVMs were done using the grid $C = 2^q$, $\gamma = 2^p$ with $q = -5, -3, \ldots, 15$; $p = -15, -13, \ldots, 3$ following [10].

4.1 Comparison of structured and non-structured sampling

In this first experiment, the way the data is sampled in the proposed method is analyzed. The proposed method samples the data with a *structure* by creating random disjoint sets in order to generate the base SVMs (line 8, Alg 1). In this experiment we compare this proposal with standard random sampling without replacement. That is, the same algorithm as the one shown in Alg 1 is used but substituting line 8 by the generation of random samples without replacement of size $tp \cdot N_{train}$ (*subbagging*). For this test the ensembles were evaluated with

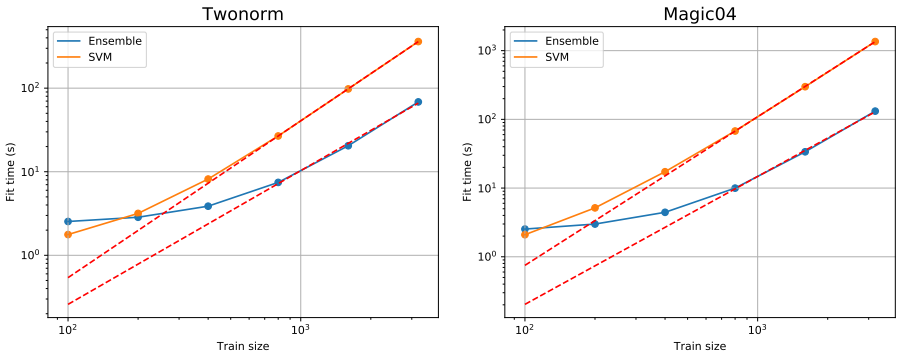

Fig. 2. Average training time. Both axis are in log scale. The red dashed line is a linear fit to the log-log data.

different values for the hyperparameter tp: [0.01, 0.02, 0.025, 0.05, 0.1, 0.2]. The hyperparameter B remained fixed for both models at $B = 10$. The mean and standard deviations reported are the average over 20 independent executions.

Figure 1 shows the results for a representation of the analyzed datasets. Similar results are obtained in the other datasets. The plots on the left column of Figure 1 display the average accuracy of the models with respect to tp with the standard deviation shown as a shaded region around the mean values. The values of the standard deviation are also represented on the plots shown on the right column.

These results show that the average accuracy achieved using the proposed method with structured sampling is more accurate than subbagging for most values of tp, although the differences are in general small. More importantly, the structured model produces results that are more stable than regular subbagging. This can be seen from the standard deviation plots where the standard deviation for subbagging is generally higher that the one of the structured model. This difference is more noticeable in general for low values of tp. In the following experiments, we will use only the proposed structured type of sampling.

4.2 Training complexity

In these experiments, we study the training time complexity for the proposed ensemble and for the single SVM as a function of the size of the training set N_{train} and as a function of hyperparameter tp. These tests have been carried out for two datasets: Magic04 and Twonorm. The reported results are averages over 10 independent executions.

In the first analysis, we study the impact of N_{train}. For the proposed ensemble, we have fixed $B = 10$ and $tp = 0.2$. For the SVM, we used 10-fold cross-validation in the grid search, i.e. $K = 10$. Both models use the same grid with $GSize = 110$ as defined above. The models have been evaluated for the following training sizes N_{train}= 100, 200, 400, 800, 1600, 3200. The limitations

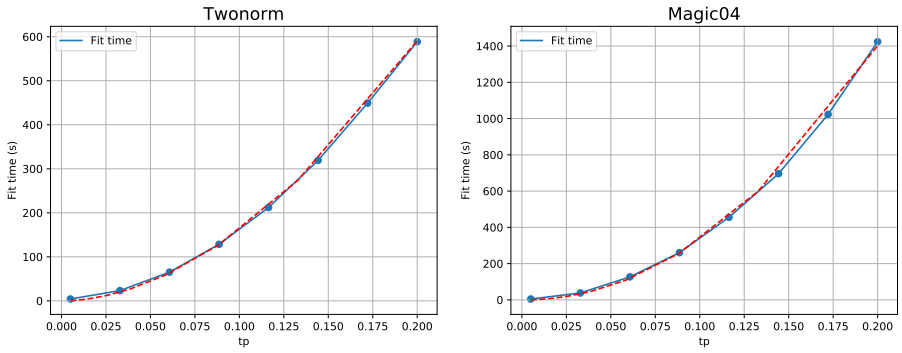

Fig. 3. Average training time of the SVM ensemble. The red dashed line indicates a linear fit to the log-log of the data.

of our workstation prevented us from using higher values of N_{train} as the cost for training a single SVM became too high. Nonetheless, the selected values provide enough data to identify the asymptotic growth of both algorithms.

The results of this experiment are presented in Figure 2 using a logarithmic scale in both axis. The plots show the execution training time in seconds with respect to N_{train} for the proposed ensemble (blue solid line) and for the single SVM trained on the whole dataset (orange solid line). Both training times include the hyperparameter search as previously described. The red dashed lines represents a linear fit to the data excluding the first 3 points in order to capture the asymptotic behavior of the sequences. The slope of these lines are 1.60, 1.88 for the ensemble and the SVM respectively in Twonorm and 1.86, 2.16 in Magic04. All four slopes show values close to 2, which indicates that the complexity of both models with respect to the training size is approximately quadratic as predicted in previous sections. It is also worth noting that there is a substantial difference between the training times in both problems. The training cost in the Magic04 dataset for training the single SVM and the SVM ensemble is, respectively, 3.7 and 1.9 times higher that their cost in Twonorm. This shows that training time for both models is very problem dependent.

In a second experiment we analyze the training time of the proposed ensemble as a function of hyperparameter tp. The parameter B and the training set size N_{train} are fixed with values $B = 10$, $N_{train} = 10000$. The ensemble is evaluated at 8 evenly spaced values of tp between 0.005 and 0.2.

The results of this experiment are presented Figure 3 with a solid blue line. In addition, the plots show with a dashed red line the linear fit to the log-log data excluding the first 3 points in order to capture the asymptotic tendency of the curve. The slopes of these lines are 1.88 in Twonorm and 2.08 in Magic04. These results show the quadratic growth of the training time with respect to tp, which confirms the cost estimation we obtained in the previous section with respect to tp (see Eq. 4).

4.3 Accuracy

In this set of experiments the relation between training time and generalization accuracy of the models is analyzed in detail. Each experiment realization involves the following steps: (i) For each of the non-synthetic problems, $N_{train} = 10000$ training instances are selected at random and the rest of the data is used as test set. For the synthetic datasets, two sets of 10000 instances are sampled randomly as training and testing sets; (ii) With the available training data, a set of ensembles of SVMs and a set of single SVMs are trained such that they require different training computational budgets. For a single SVM the most natural way of adjusting to these constraints is to reduce the number of instances used for training. In this way, the SVM uses only a fraction of the available training set. Specifically, for the Single SVM model the training subset evaluated sizes, N_{train}, are: [200, 500, 1000, 2000, 3500, 5000, 10000] for all problems except for Adult, for which the last value (10000) was not evaluated due to computational limitations of our workstation. For the ensemble, the parameters B and tp can be reduced to adjust to the constraints while using all the available data. In these tests we only modify tp and leave B fixed at a value that generally yields good results $B = 10$. For the ensemble model the evaluated tp values are: [0.01, 0.025, 0.05, 0.1, 0.2, 0.25]; (iii) Finally all trained models were evaluated in the same left out test set. The scores presented are the result of averaging 20 independent executions. The training time measurements are the result of averaging 4 executions. The reason for this difference in the number of repetitions is twofold: Firstly, time measurements are more stable and therefore increasing the number of repetitions does not improve the precision of the measurement. Secondly, measuring the training times is delicate as they must be taken in a controlled environment with a single thread executions. All time measurements are done in a single thread process in order to attain a clear over all estimation of the total computational resources needed.

Figure 4 represents the average generalization accuracy obtained by each model (vertical axis) with respect to the corresponding training time required to train the models (horizontal axis) represented in logarithmic scale. The standard deviation of the accuracy is represented by a shaded region around the mean. The numerical figures for these plots are also presented in Table 1 for the different values of tp and training size for the ensemble (left part of the table) and single SVM respectively (right part).

The main observation that we extract from these plots is that for a fixed training time budget, the ensemble consistently outperforms the single SVM. The differences between their accuracies is specially significant for the cases with limited time budgets ($< 100s$). Comparing maximum scores, regardless of the training time, we observe that the ensemble has the highest score in 3 datasets: Adult, Twonorm and Ringnorm, while the SVM has the highest score in the other 3. However, the highest scores of the SVM are always achieved for $N_{train} = 10000$, which involve very large training times (all above $10000s$). Comparing the training times in these datasets of the highest scores achieved by each model, we observe that the SVM training time is 7.6, 36.8 and 6.6 higher

Dataset	SVM Ensemble			Single SVM		
	tp	Score	Time (s)	N_{train}	Score	Time (s)
Magic04	0.010	82.82±0.63	6.40±0.20	200	80.65±1.51	4.80±0.37
	0.025	84.72±0.31	22.54±1.28	500	83.27±0.76	25.81±2.44
	0.050	85.34±0.31	77.34±0.47	1000	84.60±0.34	102.87±1.18
	0.100	85.80±0.29	306.35±3.05	2000	85.41±0.27	437.49±57.67
	0.200	86.10±0.38	1462.35±118.66	3500	85.88±0.21	1445.34±53.12
	0.250	86.30±0.23	1991.58±13.51	5000	86.14±0.20	3145.13±129.61
				10000	86.58±0.21	15127.35±514.01
Bank	0.010	88.33±0.05	7.13±0.03	200	88.03±0.90	5.68±0.11
	0.025	89.29±0.40	26.84±0.20	500	88.86±0.53	28.44±1.08
	0.050	89.89±0.20	92.52±0.33	1000	89.29±0.41	102.69±1.55
	0.100	89.93±0.18	341.74±1.84	2000	89.65±0.25	392.08±7.23
	0.200	89.83±0.27	1304.53±9.86	3500	89.72±0.24	1190.67±19.23
	0.250	89.90±0.22	2023.25±30.18	5000	89.76±0.17	2503.75±70.17
				10000	90.02±0.11	12543.61±64.57
Adult	0.010	81.35±1.22	12.51±0.10	200	80.05±1.36	10.50±0.10
	0.025	83.64±0.35	54.97±0.25	500	82.41±0.58	58.12±1.46
	0.050	84.35±0.28	199.76±0.58	1000	83.62±0.38	227.98±8.40
	0.100	84.80±0.27	774.11±6.40	2000	84.08±0.34	967.83±34.18
	0.200	84.95±0.26	3285.41±36.97	3500	84.47±0.16	3654.76±137.48
	0.250	85.07±0.29	5428.58±45.47	5000	84.69±0.15	9027.30±833.47
				10000	-	-
Twonorm	0.010	97.79±0.08	5.69±0.02	200	97.31±0.43	3.71±0.04
	0.025	97.79±0.08	16.91±0.07	500	97.52±0.15	15.73±0.08
	0.050	97.80±0.08	52.51±0.12	1000	97.53±0.12	55.07±0.82
	0.100	97.79±0.07	185.27±0.31	2000	97.60±0.09	204.13±1.66
	0.200	97.80±0.10	667.53±4.54	3500	97.62±0.06	575.05±10.08
	0.250	97.80±0.08	1008.01±7.60	5000	97.62±0.05	1156.25±10.82
				10000	97.75±0.11	4819.12±37.73
Threenorm	0.010	86.87±0.22	6.47±0.04	200	85.10±1.14	4.92±0.06
	0.025	87.13±0.14	22.15±0.07	500	86.41±0.66	22.90±0.78
	0.050	87.59±0.21	73.63±0.97	1000	87.29±0.44	107.42±7.76
	0.100	88.14±0.22	287.44±3.79	2000	87.66±0.43	403.86±34.77
	0.200	88.57±0.14	1142.24±19.12	3500	88.32±0.20	1159.19±17.81
	0.250	88.58±0.16	1806.76±16.54	5000	88.46±0.22	2456.30±48.17
				10000	88.78±0.18	11923.65±273.75
Ringnorm	0.010	98.47±0.14	6.60±0.03	200	98.24±0.15	5.81±0.07
	0.025	98.54±0.12	20.44±0.06	500	98.30±0.25	21.06±2.08
	0.050	98.53±0.12	63.19±0.27	1000	98.28±0.30	69.25±0.59
	0.100	98.52±0.15	223.34±2.53	2000	98.46±0.15	251.93±0.49
	0.200	98.51±0.11	808.90±3.21	3500	98.49±0.12	733.09±8.12
	0.250	98.51±0.12	1238.11±3.64	5000	98.49±0.16	1459.79±8.94
				10000	98.51±0.09	6163.33±75.59

Table 1. Average scores and training times of Ensemble and Single SVM.

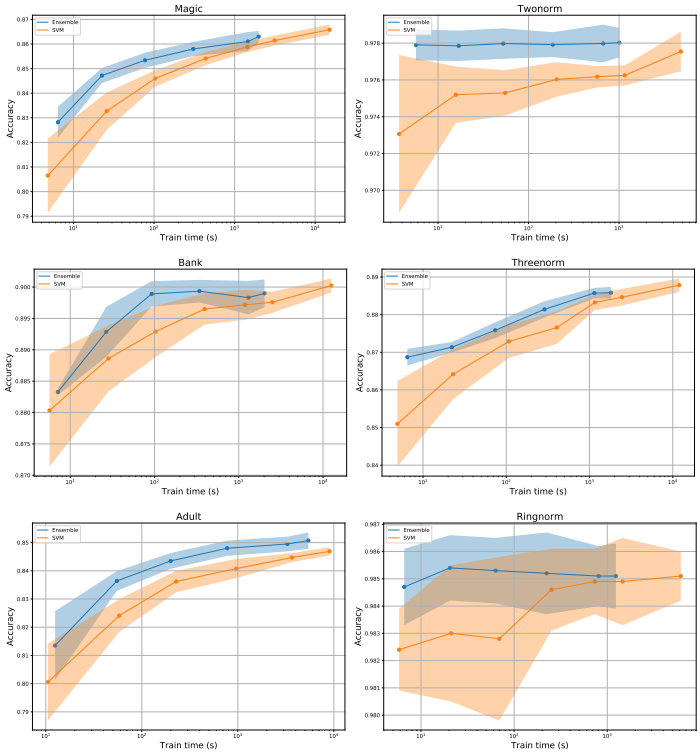

Fig. 4. Accuracy comparison between Structured Subbagging and single SVM. The vertical axis represents the score, the horizontal axis represents the training time in logarithmic scale. The shaded region represents the standard deviation of the scores.

that the time for the best accuracy of the ensemble for Magic04, Bank and Threenorm respectively (see Table 1). In any case, the difference between the best accuracies of each method in the different datasets are rather small (except maybe in Adult in which we could not run the $N_{train} = 10000$ for the single SVM) with a clear advantage of the proposed ensemble in terms of training speed.

Another interesting observation that can be made from Figure 4 is that in four out of the six problems the accuracy of the ensemble improves when the value of tp increases. In the other two datasets, Twonorm and Ringnorm, the ensemble achieves a similar accuracy for all values of tp. This tendency to achieve better accuracies for higher values of tp indicates that the accuracy of the ensemble benefits from training the base models on larger training sets despite of the fact that less base models are combined. Note that, in this experiment, ensembles for all values of tp have converged before reaching the given number for combined combined models. Hence, the differences in accuracy are not caused by lack of convergence of the models.

5 Conclusions

In this paper a novel ensemble of SVMs is presented. The proposed model creates a diverse set of simple SVMs using disjoint subsets of the training set and using different hyper-parameters. We show that the proposed method is more efficient to train computationally than a single SVM trained on the complete dataset. In addition the computational cost can be tuned using two hyper-parameters of the model: the number of subensembles and the fraction of training instances to use for each SVM of the ensemble. We carried out an extensive analysis of the model with respect to these hyper-parameters that showed the generalization accuracy of the model for any given computational budget is better than that of a single SVM.

Acknowledgments

This work was supported by PID2019-106827GB-I00 / AEI / 10.13039/501100011033

References

1. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory. pp. 144–152. COLT '92 (1992)
2. Breiman, L.: Bias, variance, and arcing classifiers. Tech. Rep. 460, Statistics Department, University of California (1996)
3. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology **2**, 27:1–27:27 (2011)
4. Cherkassky, V., Ma, Y.: Practical selection of svm parameters and noise estimation for svm regression. Neural Networks **17**(1), 113 – 126 (2004)
5. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**(3), 273–297 (Sep 1995)
6. Dua, D., Graff, C.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml
7. Kashef, R.: A boosted svm classifier trained by incremental learning and decremental unlearning approach. Expert Systems with Applications **167** (2021)
8. Mayhua-López, E., Gómez-Verdejo, V., Figueiras-Vidal, A.R.: A new boosting design of support vector machine classifiers. Information Fusion **25** (2015)
9. Moro, S., Cortez, P., Rita, P.: A data-driven approach to predict the success of bank telemarketing. Decision Support Systems **62**, 22 – 31 (2014)
10. Sabzevari, M., Martínez-Muñoz, G., Suárez, A.: Randomization vs optimization in svm ensembles. In: 27th ICANN. pp. 415–421 (09 2018)
11. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Proc. of the 25th NIPS. p. 2951–2959 (2012)
12. Stork, J., Ramos, R., Koch, P., Konen, W.: Svm ensembles are better when different kernel types are combined. In: Data Science, Learning by Latent Structures, and Knowledge Discovery. pp. 191–201 (2015)