Subsampling strategies in SVM ensembles

Patrick Koch and Wolfgang Konen

Cologne University of Applied Sciences
Institute of Computer Science
E-Mail: {patrick.koch, wolfgang.konen}@fh-koeln.de

Abstract

Support Vector Machines (SVMs) have shown to be strong methods for classification problems. Especially for difficult tasks the performance of SVMs is often superior to other learning algorithms. A main issue arising with this kernel-based learning is the high computation time and also the large memory demand required for training with large data. As a solution to this, ensemble-based SVM approaches have recently been proposed. Meyer *et al.* [1] investigated SVM ensembles based on bagging [2] and Cascade SVMs [3]. Stork *et al.* [4, 5] proposed ensembles based on boosting [6] and bagging with subsampling of the training data. In their experimental study they observed that subsampling is a necessary ingredient to impede overfitting. Unfortunately no rule-of-thumb could be given for the sample size parameter. The goal of this study is to get a deeper understanding which elements in a fruitful combination of individuals in SVM ensembles lead to considerable time savings while maintaining a good classification accuracy. First, we expect to obtain an asymptotic behaviour when we increase the ensemble size for a fixed training set size setting. Secondly, we want to measure the influence of the training set size on the classification accuracy. With these findings we try to give recommendations for sample size and ensemble size in order to balance computation time and accuracy. As a nice side effect, the observations made in this study can be used to create ensembles of other learning algorithms as well.

1 Introduction

Support Vector Machines (SVMs) [7, 8, 9] are state-of-the-art learning algorithms for supervised machine learning. SVMs often have shown superior performances both in classification and regression. A drawback of the method is that it badly scales with increasing dataset sizes. This can be a problem today, as dataset sizes are more and more increasing.

Recently, alternatives based on ensemble learning have been proposed to circumvent this issue. E.g., Meyer *et al.* [1] investigated SVM ensembles based on bagging [2] and Cascade SVMs [3]. Stork *et al.* [4, 5] proposed ensembles based on boosting [6] and bagging with subsampling of the training data.

In this study we analyze the influence of the ensemble size (number of iterations in boosting) related to the sample size inside the boosted learners. By doing this we expect to find an asymptotic behaviour of the ensemble learner, e.g., the classification performance should increase with larger ensemble sizes, but finally should converge when no further improvements are possible.

The paper is structured as follows: we give a brief overview about Support Vector Machines in Sec. 2.1 and the boosting algorithm (AdaBoost) in Sec. 2.2. In Sec. 3 we perform an experimental analysis on datasets from the public UCI repository [10]. The results are discussed in Sec. 4 and we summarize with a conclusion and outlook on future research in Sec. 5.

2 Methods

2.1 Support Vector Machines

Support Vector Machines (SVMs) [7, 8, 9] are state-of-the-art learning algorithms for classification and regression. In classification, data can be written as a number of n observations

$$(\vec{x}_1, y_1), (\vec{x}_2, y_2), ..., (\vec{x}_n, y_n) \in \mathcal{X} \times \mathcal{Y} \tag{1}$$

where $x_i \in \mathcal{X}$ are the input patterns and $y_i \in \mathcal{Y}$ the corresponding class labels for pattern x_i. In the simplest form, the output set $\mathcal{Y}$ only contains two elements, leading to binary classification, where the classes are often denoted by $\mathcal{Y} = \{-1, 1\}$

For linearly separable data, SVM fit a linear classifier, maximizing the margin between the classes in order to give the best generalization performance. But as data is often not linearly separable, it is the core of machine learning that two observations "being near in input space" should have a similar output value. Therefore, SVM incorporate kernel functions

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \tag{2}$$

denoting the similarity of two observations. The kernel function k needs to suffice several condtitions, e.g., symmetry, and positive semi-definiteness. The function itself can be interpreted as a dot product in a high-dimensional space [11]. It enhances the SVM learning algorithm by an implicit mapping of the input data into a higher-dimensional feature space, where a linear classifier is applicable. In this paper, without loss of generality we use only one single kernel function, that is the radial basis kernel

$$k(\vec{x}, \vec{z}) = \exp(\gamma \cdot ||\vec{x} - \vec{z}||^2) \tag{3}$$

with hyperparameter $\gamma \in \mathbb{R}^+$. For a comparison with other kernels on the same task see [4, 5].

An optimal prediction model can now be determined by introducing the associated reproducing kernel Hilbert space H for the kernel function k and solving the optimization problem:

$$\hat{f} = \arg \inf_{f \in H, b \in \mathbb{R}} ||f||_H^2 + C \sum_{i=1}^{n} L(y_i, f(\vec{x}_i) + b) \ . \tag{4}$$

The first summand $||f||_H^2$ defines a penalty and in case of the 2-norm penalizes non-smooth functions. Because the function f maps into $\mathbb{R}$, the sign is calculated in the case of binary classification. Finally the second term measures the closeness of the predictions to the true outputs. The closeness is defined by a loss function, that is usually the Hinge loss $L(y,t) = L_h(y,t) = \max(0, 1 - yt)$ in case of classification. The Hinge loss is a convex, upper surrogate loss for the 0/1-loss (which is a desired loss function, but algorithmically intractable). A hyperparameter C controls the balance between the smoothness and the loss function.

SVM are ideally suited for binary classification tasks, but can also handle more classes. Approaches for multi-class problems have been proposed by Weston and Watkins [12], and Crammer and Singer [13] gave an alternative formulation.

2.2 AdaBoost

AdaBoost, as a shorthand for Adaptive Boosting, was proposed in 1995 by Freund and Schapire [6]. The basic AdaBoost algorithm is shown in Algorithm 1. It works by repeatedly building and evaluating weak classifiers on the training set where each time a different sample from the training set distribution is drawn. Misclassified records in previous iterations get higher weights, leading to a stronger emphasis on those records by the forthcoming classifiers. For each classifier h_t its quality $\alpha_t \in [0, \infty]$ on the original training set is evalulated. The final ensemble output is that class with the largest sum of α_t, where the sum is calculated for all classifiers voting for that class.

Algorithm 1 Multi-class SVM AdaBoost algorithm

```
1: Input: a training set $\Gamma = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_N, y_N)\}$ with class labels $y_i$ having $K$ levels.
2: Initialize: the weights $w_i^1 = 1/N$ for $i = 1, \ldots, N$.
3: Define $\Theta(P) = 1$ if $P$ is true, 0 else.
4: for ($t = 1, \ldots, T$) do
5:     Draw a $w_i^t$-weighted training sample set $S$ of size $s = bN$ with replacement from $\Gamma$, where $b \leq 1$ denotes a fraction of the training set.
6:     Train a weak learner $h_t$ on $S$. Here $h_t$ is a SVM.
7:     Calculate training error $\epsilon_t = \sum_{i=1}^{N} w_i \, \Theta(h_t(\vec{x}_i) \neq y_i)$ on set $\Gamma$.
8:     Set $\alpha_t = \frac{1}{2}\left(ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) + ln(K-1)\right)$ (quality of weak learner $h_t$).
9:     Update weights: $w_i^{t+1} = w_i^t exp(\alpha_t \Theta(h_t(\vec{x}_i) \neq y_i))/Z$, where $Z$ is a normalization constant such that $\sum_{i=1}^{N} w_i^{t+1} = 1$.
10: end for
11: Output: $f(\vec{x}) = arg\ \max_c \left(\sum_{t=1}^{T} \alpha_t \Theta(h_t(\vec{x}) = c)\right)$.
```

Compared to the original AdaBoost algorithm, our SVM AdaBoost with SVMs as the base classifiers has three modifications:

1. As Wickramaratna et al. [14] pointed out, it is essential for the classifiers to be *weak* in order to make AdaBoost productive. Since SVMs tend to be strong classifiers, it is necessary to weaken them. We sample in S for each classifier only a small fraction b of the set Γ in Algorithm 1, Step 5, e.g. $b = 0.1$ or $b = 0.01$. Note that the evaluation in Step 7 and weight update in Step 9 is done on the <u>whole</u> set Γ: This gives a precise figure of merit for each classifier and keeps the weights in sync. Training on the set S with only $s = bN$ records has the nice side effect that we can tackle large datasets with SVM, without being blocked by runtimes increasing approximately cubically with the number of training records.

2. As a measure to increase the diversity of the ensemble, we choose for the radial SVMs the width γ randomly and uniformly from the 0.1 to 0.9 quantile range of $|\vec{x} - \vec{x}'|^2$, as suggested in [15], where x, x' are distinct data points. We found that this gives a better ensemble performance than using a tuned but fixed γ.

3. Initially we use the full training set of size N_{full} (see Tab. 1) for the set Γ. Later in Sec. 3.3 we experiment with subsamples Γ of size $N < N_{full}$, which are smaller than the full training set.

3 Experimental analysis

Earlier experiments [4, 5] showed that for arbitrary kernel types better results could be obtained with SVM AdaBoost, when the sample size $s = bN$ during training is fixed at a small level. In cases where the sample size was too large, the algorithm showed a remarkable overfitting on several benchmarks [4, 5]. This situation is shown exemplarily for dataset Adult from the UCI repository in Fig. 1. Here the performance of small and large sample sizes together with different kernel types is compared and it can be seen that small sample sizes ($b \in [0.05, 0.12]$) are better for all kernels.

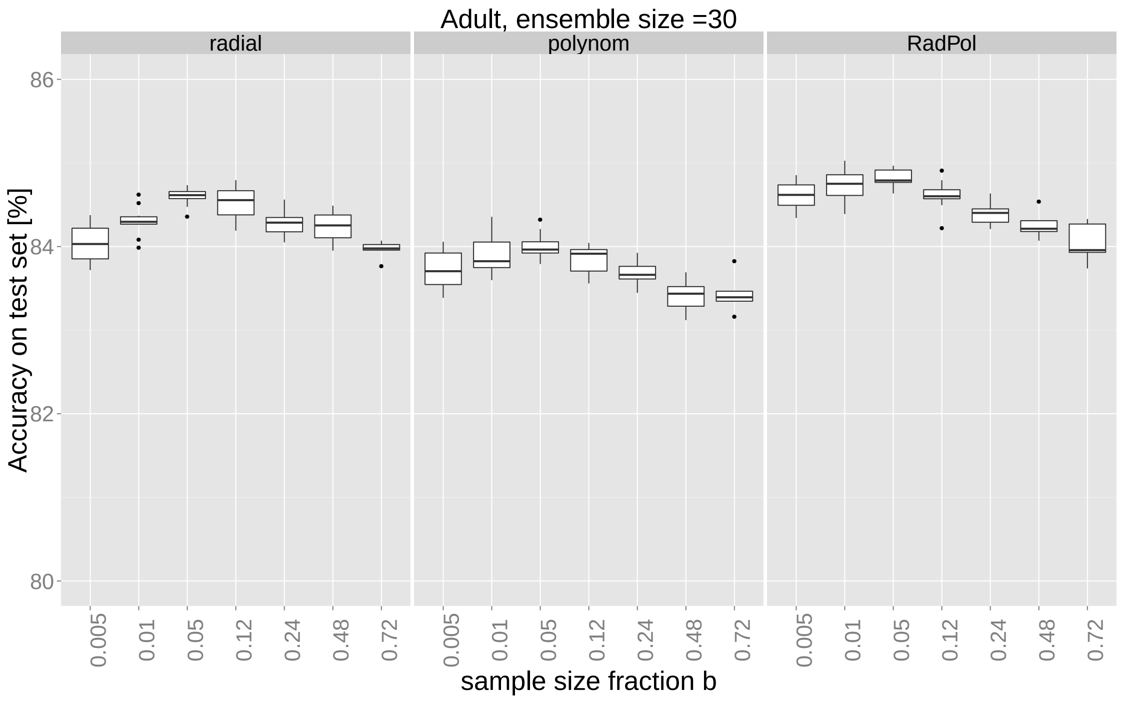

Figure 1: Test set accuracy on task Adult with SVM AdaBoost. In case of training with larger sample size fractions than $b = 0.12$, the ensemble suffers from considerable overfitting. The optimal fraction is in this case in the range $b \in [0.05, 0.12]$. Mixed ensembles like “RadPol” =(radial + polynomial kernels) are better than a pure “radial” ensemble.

We observed also in some earlier experiments [4, 5] a steady increase with increasing ensemble size, but only ensembles up to $T = 50$ were investigated at that time. Therefore it seems natural to ask whether a further increase in ensemble size can boost the performance even more or whether a converging behaviour is observed. We perform the relevant experiment in Sec. 3.2.

Besides the better performance, it is notable that large speed-ups are possible when training with small sample sizes inside the ensemble. For this reason we have set the sample size fraction to very low values, e.g. $b = 0.5\%$

Table 1: Datasets used in experiments: Number of records, training records (N_{full}), number of features and number of classes. The test set is the set of all non-training records. The dataset Acoustic$_2$ is a binarized version (class 3 vs. rest") of the Acoustic$_2$ dataset in the UCI repository [10], which is originally a three-class problem.

Name	Records	Train	Features	Classes
Adult	45222	30162	14	2
Acoustic$_2$	98528	78823	50	2

(cf. Sec. 3.2). However, for each SVM in the ensemble, a *different* sample is drawn from the training set. It is matter of future research which sample size is suited best for a new task. Up to now this parameter was always chosen manually by the experimenters. In the experiments in Sec. 3.3 we analyze the behaviour of different training set sizes N for two benchmark datasets.

3.1 Experimental Setup

We perform an experimental study on two datasets from the UCI repository [10]. Intentionally, we select two larger-sized datasets from the repository, since in earlier studies [4, 5] SVM ensembles could especially diminish the required training times for larger datasets. Furthermore we only perform experiments on binary classification problems, since SVMs have been originally proposed for such type of problems. Nevertheless, the SVM AdaBoost algorithm can be applied to multi-class problems as well. This is true if the underlying SVM does support multiple classes, e.g., by using strategies as proposed in [12, 13].

Our experimental setup runs the SVM AdaBoost algorithm ten times with different random seeds and varying ensemble sizes (Sec. 3.2) or training set sizes (Sec. 3.3).

3.2 Results Ensemble Size

In this experiment we vary the number of iterations (ensemble size) used in AdaBoost, while keeping the sample size on a very low level (subsamples of 0.5% of the total number of training patterns). We use ensemble size settings of {10,20,30,40,50,100,200,300,400,500,600,700,800,900,1000}. Although 1000 is a very large ensemble size for SVM AdaBoost, we expect that due to the subsampling, the time spent in this experiment should still be feasible with small sample size settings in AdaBoost.

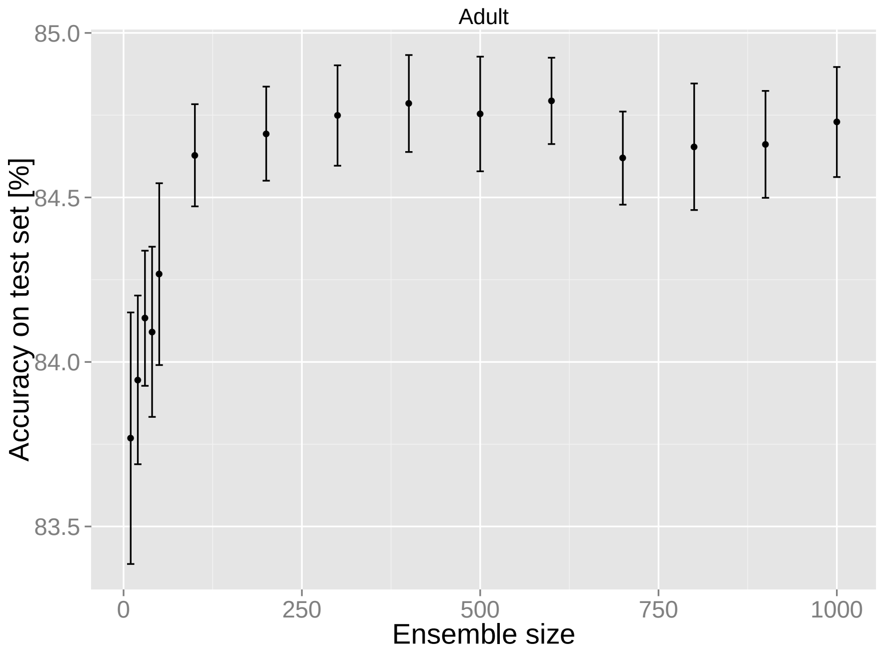

Figure 2: Mean accuracy and standard deviation of ten runs on independent test data for the "Adult" dataset. The ensemble size is varied between 10 and 1000 iterations, while keeping the sample size fixed at 0.5% at the same time.

The results illustrate the mean classification accuracy plus standard deviation on independent test data for each ensemble size setting. In Fig. 2 we show the results for the Adult dataset, while in Fig. 3 we show the results for the Acoustic$_2$ dataset.

A mainly converging behaviour is seen for the Adult dataset. Here, the performance increases remarkably for the first 200 iterations of the algorithm, then fluctuates for some time at a level of 84.75%, but finally decreases slightly for ≥ 700 iterations. After the decrease the results seem to fluctuate slightly around the new level.

A similar behaviour can be observed for the Acoustic$_2$ dataset. When the ensemble size tends to be very small (only 10 or 20 iterations), we obtain a significantly inferior result compared to more iterations. This result can be reasoned due to the small sample size of only 0.5% together with only a few iterations of AdaBoost. However, when the ensemble size is set to a sufficient level (≥ 200 iterations as in the case of "Adult"), we can expect a better performance of the algorithm. For larger ensemble sizes the improvement effect is lost and similar to the results of the Adult experiment we can observe a fluctuating performance.

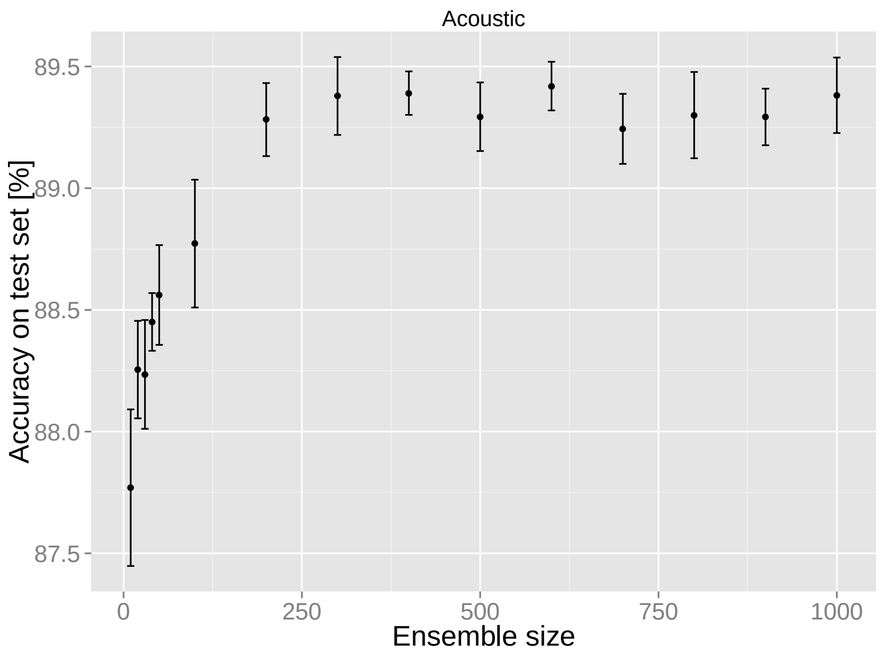

Figure 3: Mean accuracy and standard deviation of ten runs on independent test data for the "Acoustic$_2$" dataset. The ensemble size is varied between 10 and 1000 iterations, while keeping the sample size fixed at 0.5% at the same time.

3.3 Results Training Set Size

We noticed when training AdaBoost on larger datasets, that the majority (90%) of the time was *not* spent in the training of the individual SVMs h_t (Algorithm 1, Step 5) but instead it was spent in predicting $h_t(\vec{x}_i)$ for all training set records (Step 7). This is understandable since the original training set size ($N_{full} = 30162$ or 78823, see Tab. 1) is much larger than the sample size $s = 300$ used for each individual SVM.

Is it possible to reduce the total computation time by decreasing the training set size N in Algorithm 1 without loss in accuracy? We investigated this question with the following experiment: Instead of the full training set N_{full} we draw a sample of size $N = \{10\%, 25\%, 50\%, 100\%\} \times N_{full}$ and perform Algorithm 1 on that N. The sample size s was kept constant.[1]

Each experiment was repeated 10 times with different samples of size N being drawn. Fig. 4 shows the resulting accuracies on the test set. We observe that for $N = 50\% N_{full}$ there is virtually no degredation in accuracy. The smaller training set sizes $N = 25\% N_{full}$ or $N = 10\% N_{full}$ show some degradation in accuracy for Adult and nearly none for Acoustic$_2$. At the same time Tab. 2 shows, that the total computation time is diminished

[1]More precisely: $s = 1\% N_{full} = 302$ for dataset Adult and $s = 0.6\% N_{full} = 473$ for Acoustic$_2$.

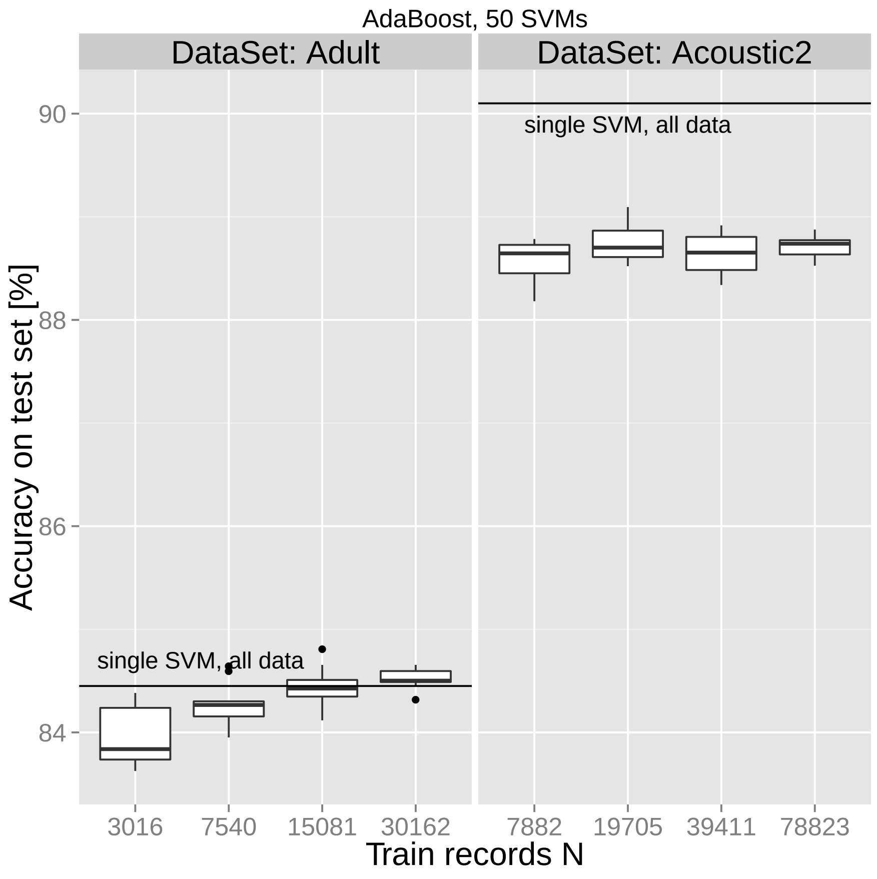

Figure 4: Boxplots: classification accuracies for our modified AdaBoost algorithm as a function of the training set size N. Horizontal lines: for comparison the accuracy of a single SVM trained on all data. We observe only a slight degradation with decreasing N. At the same time, the computation time decreases largely, see Tab. 2.

by a factor of approximately 2, 4 or 8 for $N = (50\%, 25\%, 10\%) \times N_{full}$, resp., as compared to $N = N_{full}$.

4 Discussion

4.1 Stopping Rule Ensemble Size

The converging behaviour of the accuracy with increasing ensemble sizes can be used to define a stopping criterion. A possible measure is

$$\frac{\Delta A/A}{\Delta T} < \theta \tag{5}$$

where $\Delta A/A$ is the relative change in accuracy on the test set, ΔT denotes the change in ensemble size when moving from one point to the next in

Table 2: Computation times in seconds for the AdaBoost algorithm on larger datasets. The training set size N varies, while the sample set size s is kept constant for each dataset. The ensemble size is 50. Column 'speedup' shows the speedup factor as compared to a single SVM trained on all data (last line). Especially for the bigger dataset Acoustic$_2$, impressive speedups can be achieved.

Adult				Acoustic$_2$			
N	s	time [s]	speedup	N	s	time [s]	speedup
3016	302	8.8	16.6	7882	473	35.2	109.7
7540	302	17.1	8.6	19705	473	80.9	47.7
15081	302	31.8	4.6	39411	473	160.8	24.0
30162	302	61.8	2.3	78823	473	322.8	11.9
1-SVM		148.2	1.0	1-SVM		3866.0	1.0

Fig. 2 or Fig. 3, and θ is a suitable threshold. ΔT should be at least 20 or higher to accumulate enough statistical evidence.

If we set $\theta = 10^{-6}$ in our experiments, then the stopping point would be $T = 200$ for dataset Adult and $T = 400$ for Acoustic$_2$. If we set $\theta = 0$, thus detecting the first turning point, then the stopping point prior to the sign change of the measure would be $T = 400$ for both datasets Adult and Acoustic$_2$. Both results reflect quite well the convergence to a plateau in Figs. 2 and 3.

A similar stopping rule can be set up for the sample size s. Further work is needed to set up and test such a rule on a variety of benchmarks.

4.2 Related Work

Meyer *et al.* [1] analyzed an approach named *Cascade SVM*, that is likewise a sequential ensemble method. Cascade SVM aims at reducing the training time by splitting the training data into k disjoint subsets. For each of these subsets a single SVM is trained. The support vectors of the single SVMs then constitute the training set in the subsequent steps. By doing this, the training size is kept low, but the subsequent SVMs still receive the support vectors of the precedent classifiers. Meyer *et al.* [1] observed that the Cascade SVM performs almost equal to a single SVM on a variety of datasets, but the method is much slower in computation than an approach based on Bagging, that is also analyzed by Stork *et al.* [4, 5].

The AdaBoost approach described in this paper is considerably different from the Cascade SVM investigated by Meyer *et al.* [1]. AdaBoost is also sequential, but introduces weights for the training patterns. It is not the case that patterns are 'thrown away' as a consequence of earlier steps. For this reason it is not necessary to perform multiple runs of the algorithm to reach a reasonable and non-overfitting classifier. Instead it could be shown in this article that SVM AdaBoost tends to show an asymptotic and robust behaviour, even when only small training set sizes are encountered.

5 Conclusions and Outlook

We performed experiments using AdaBoost as an ensemble strategy build upon SVMs where each individual learner in the ensemple needs only a small sample of the full training set. With a very small sample size (only 0.5% of the total data available for training) and only a few iterations of the algorithm we obtain classification results slightly inferior to a single SVM trained on all data and we see a high variance of these results. But with more iterations of the algorithm (larger ensemble sizes) the variance in the results decreases and the classification accuracy comes close to the single-SVM-all-data accuracy or even surpasses it. The accuracy converges in a plateau when the number of iterations is further increased.

For our medium-sized datasets SVM AdaBoost is faster than a single SVM trained on all data. Thus, the SVM AdaBoost algorithm constitutes a viable alternative for applying SVM to even larger datasets where a single SVM cannot handle all data in reasonable time.

In another experiment we varied the training set size N, while keeping the ensemble size and the sample size s fixed at the same time. Surprisingly, the ensemble method did not perform worse with about 50% of the total training data for the Adult dataset, and for the Acoustic$_2$ dataset even smaller sample sizes of only 25% were found to be sufficient. In both cases the classification accuracy with the subsampled training data was only slightly worse compared to the full training data. At the same time we observe large speed-ups in the total training time of the SVM AdaBoost algorithm.

We proposed with Eq. (5) a stopping rule to find the right ensemble size in SVM AdaBoost for a given dataset. In future work we want to apply this stopping rule to further datasets to test its general validity and to formulate and test a similar stopping rule for finding semi-automatically the right sample size as well.

Literatur

[1] Meyer, O.; Bischl, B.; Weihs, C.: Support Vector Machines on Large Data Sets: Simple Parallel Approaches. In: *Data Analysis, Machine Learning and Knowledge Discovery* (Spiliopoulou, M.; et al., Hg.). Springer. 2013.

[2] Breiman, L.: Bagging predictors. *Machine Learning* 24 (1996) 2, S. 123–140.

[3] Graf, H.; Cosatto, E.; Bottou, L.; Dourdanovic, I.; Vapnik, V.: Parallel Support Vector Machines: The Cascade SVM. *Advances in Neural Information Processing Systems* 17 (2004), S. 521–528.

[4] Stork, J.; Ramos, R.: Case Study Report: Building and analyzing SVM ensembles with Bagging and AdaBoost on big data sets. CIOP Technical Report 04/2013, Cologne University of Applied Sciences. URL `http://gociop.de/ciop-reports`. 2013.

[5] Stork, J.; Ramos, R.; Koch, P.; Konen, W.: SVM ensembles are better when different kernel types are combined. *Studies in Classification, Data Analysis, and Knowledge Organization* (2014), S. 1–10. In preparation.

[6] Freund, Y.; Schapire, R.: A decision-theoretic generalization of on-line learning and an application to boosting. In: *Proceedings of the Second European Conference on Computational Learning Theory*, EuroCOLT ’95, S. 23–37. London, UK. ISBN 3-540-59119-2. 1995.

[7] Cortes, C.; Vapnik, V.: Support Vector Machine. *Machine Learning* 20 (1995) 3, S. 273–297.

[8] Schölkopf, B.; Smola, A.: *Learning with kernels: Support Vector Machines, regularization, optimization and beyond*. MIT Press. 2002.

[9] Cristianini, N.; Shawe-Taylor, J.: Support Vector Machines. 2000.

[10] Bache, K.; Lichman, M.: UCI Machine Learning Repository. URL `http://archive.ics.uci.edu/ml`. 2013.

[11] Mercer, J.: Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London* 209 (1909), S. 415–446.

[12] Weston, J.; Watkins, C.: Support Vector Machines for multi-class pattern recognition. In: *Proceedings of the 7th European Symposium on Artificial Neural Networks (ESANN)*, Bd. 99, S. 61–72. 1999.

[13] Crammer, K.; Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research* 2 (2002), S. 265–292.

[14] Wickramaratna, J.; Holden, S. B.; Buxton, B.: Performance Degradation in Boosting. In: *Proceedings of the 2nd International Workshop on Multiple Classifier Systems* (Kittler, J.; Roli, F., Hg.), Nr. 2096 in LNCS, S. 11–21. Cambridge, UK. 2001.

[15] Caputo, B.; Sim, K.; Furesjo, F.; Smola, A.: Appearance-based Object Recognition using SVMs: Which Kernel Should I Use? *Proceedings of NIPS Workshop on Statistical Methods for Computational Experiments in Visual Processing and Computer Vision* (2002).