SVM ensembles are better when different kernel types are combined

Jörg Stork, Ricardo Ramos, Patrick Koch, and Wolfgang Konen

Abstract Support Vector Machines (SVM) are strong classifiers, but large data sets might lead to prohibitively long computation times and high memory requirements. SVM ensembles, where each single SVM sees only a fraction of the data, can be an approach to overcome this barrier. In continuation of related work in this field we construct SVM ensembles with Bagging and Boosting. As a new idea we analyze SVM ensembles with different kernel types (linear, polynomial, RBF) involved inside the ensemble. The goal is to train *one* strong SVM ensemble classifier for large data sets with less time and memory requirements than a single SVM on all data. From our experiments we find evidence for the following facts: Combining different kernel types can lead to an ensemble classifier stronger than each individual SVM on all training data and stronger than ensembles from a single kernel type alone. Boosting is only productive if we make each single SVM sufficiently weak, otherwise we observe overfitting. Even for very small training sample sizes – and thus greatly reduced time and memory requirements – the ensemble approach often delivers accuracies close to or better than a single SVM trained on all data.

1 Introduction

1.1 Related work

Several researchers have studied SVM ensembles during the last years (see e.g. Wang et al. [13] and references therein), predominantly in an attempt to strengthen the overall accuracy, not with the large data aspect in focus. Yu et al. [16] and Chang et al. [4] present different approaches to tackle the large data aspect, like cluster-based data selection or parallelization, but they do not use SVM ensembles.

Jörg Stork · Ricardo Ramos · Patrick Koch · Wolfgang Konen
Cologne University of Applied Sciences, e-mail: wolfgang.konen@fh-koeln.de

Recently, Meyer et al. [11] analyzed two variants of SVM ensembles for large data sets, which were based on Bagging and Cascade SVM.

1.2 Research questions

We analyze the following hypotheses in this article:

H-1 An ensemble classifier combining different kernel types performs better than ensembles from a single kernel type alone.
H-2 Boosting is only productive if we make each single SVM sufficiently weak, otherwise we observe overfitting.
H-3 Even for very small training sample sizes – and thus greatly reduced time and memory requirements – the ensemble approach often delivers accuracies close to or better than a single SVM trained on all data.

2 Methods

We give a brief introduction to Support Vector Machines in Sec. 2.1 for classification tasks. Two well-known ensemble methods are incorporated in the experiments: the Bagging approach is introduced in Sec. 2.2, whereas the AdaBoost approach is delineated in Sec. 2.3.

2.1 SVM

SVM [5, 12, 7] are state-of-the-art learning algorithms for classification and regression. In classification, data is usually written as a number of n observations

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

where the set $\mathscr{X}$ defines the input values describing the patterns and the set $\mathscr{Y}$ comprises the corresponding class labels. In the simplest form, the output set only contains two elements, leading to binary classification, where the classes are often denoted by $\mathscr{Y} = \{-1, 1\}$

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

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

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 [10]. 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 our experiments we incorporate a selection of the most commonly used kernel functions

linear:	$k(\mathbf{x},\mathbf{z}) = \mathbf{x}^T \cdot \mathbf{z}$
polynomial:	$k(\mathbf{x},\mathbf{z}) = (\mathbf{x}^T \cdot \mathbf{z} + c_0)^d$
radial:	$k(\mathbf{x},\mathbf{z}) = \exp(\gamma \cdot \|\|\mathbf{x}-\mathbf{z}\|\|^2)$

where γ, c_0 and d are hyperparameters of the corresponding functions.

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:

$$\hat{f} = \arg \inf_{f \in H, b \in \mathbb{R}} ||f||_H^2 + C \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i) + b) \ . \tag{3}$$

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 [14], and Crammer and Singer [6] gave an alternative formulation.

2.2 Bagging

Bagging [1], as a shorthand for bootstrap aggregation, is a well-known meta-algorithm to improve base classifiers in terms of stability and accuracy. The underlying idea is simple: Form several bootstrap samples by uniformly sampling T records with replacement from the full training set with N records ($T \leq N$). Sub-classifiers are trained on each of these bootstrap samples, and the final classifier makes its prediction by aggregating the predictions of all sub-classifiers. Typical aggregation methods are:

Majority voting Predict the class most often predicted by the sub-classifiers (ties broken randomly).

Probability sum If each sub-classifier delivers class probabilities, sum them up and predict the class with the highest probability sum.

Algorithm 1 Basic multi-class AdaBoost algorithm. See text for our modifications.

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

Our Bagging approach has SVMs as sub-classifiers and investigates the benefits of combining different kernel types in one ensemble. Besides the pure types Lin (linear), Rad (radial, RBF) and Pol (polynomial) we form also mixed ensembles

LinPol	linear + polynomial
RadPol	radial + polynomial
LinRad	linear + radial
LinRadPol	linear + radial + polynomial

E.g., given three ensembles Lin, Rad, Pol of ensemble size 10 each, a mixed ensemble LinRadPol of size 30 is built by joining these three.

2.3 AdaBoost

AdaBoost, as a shorthand for Adaptive Boosting, was formulated 1995 by Freund and Schapire [9]. 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 concentration for these records by the fortcoming 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.

When applying AdaBoost to SVM as the base classifier, we propose three modifications

1. As Wickramaratna et al. [15] 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 4., e.g. $b = 0.1$ or $b = 0.01$. Note that the evaluation in Step 6. and weight update in Step 8. is done on the

Table 1 Datasets used in experiments: Number of records, number of training records, number of features and number of classes

Name	Records	Train	Features	Classes	Remarks
Spam	4601	3036	57	2	
OptDigit	5620	3823	64	10	
Satellite	6435	4435	36	6	
Adult	45222	30162	14	2	
Acoustic	98528	78823	50	3	
Acoustic2	98528	78823	50	2	class 3 vs. rest

full 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 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. To increase the diversity of the ensemble, we combine the results from different kernel types. We consider here the three well known SVM kernel types radial (RBF), polynomial and linear. Two alternative ensemble-forming methods are proposed:

   Mixed In each iteration, one of the three types is selected at random to be the next weak classifier.

   Combined First, three sub-ensembles of pure type (radial, polynomial or linear) are formed, using the basic AdaBoost algorithm. The combined ensemble is formed by taking all classifiers h_t with their individual α_t from the three sub-ensembles. This ensemble predicts in the usual way the output on new cases.

3. As a further measure to increase diversity we choose for the radial SVMs in the ensemble the width γ randomly and uniformly from the .1 to .9 quantile range of $|\mathbf{x}-\mathbf{x}'|^2$, as suggested in [2], where x,x' are distinct data points. We found that this gives a better ensemble performance than using a tuned but fixed γ.

3 Experiment setup

We tested our methods on several medium-sized machine learning datasets with $3,000$ to $98,000$ records from the UCI repository. Their characteristics are given in Table 1. In the cases where the original dataset provides a separation in training and test set, we used it in our experiments (column Train). Otherwise the dataset was randomly split in 2/3 training and 1/3 test data.

To compare accuracy and speedup we have run the following algorithms on the datasets: The basic SVM, modified AdaBoost-SVM with pure, mixed and combined ensembles (see Sec. 2.3) and a reduced training fraction (parameter b), Bagging-

SVM with pure and mixed ensembles (Rad, RadPol, LinRadPol) and probability-based prediction, Bagging-SVM with the same ensembles and majority voting. For the underlying SVM we used the R package `e1071` [8] which is based on the popular LIBSVM implementation [3]. All algorithms were run repeatedly (10 times) with different training samples. The mean classification accuracy and its standard deviation is reported, as well as the training time on a single machine.

We performed a basic tuning of the SVM hpyerparameters: The parameters c_0 and d were tuned by grid search and the parameter γ was choosen according to the recipe of Sec. 2.3, item 3.

4 Results

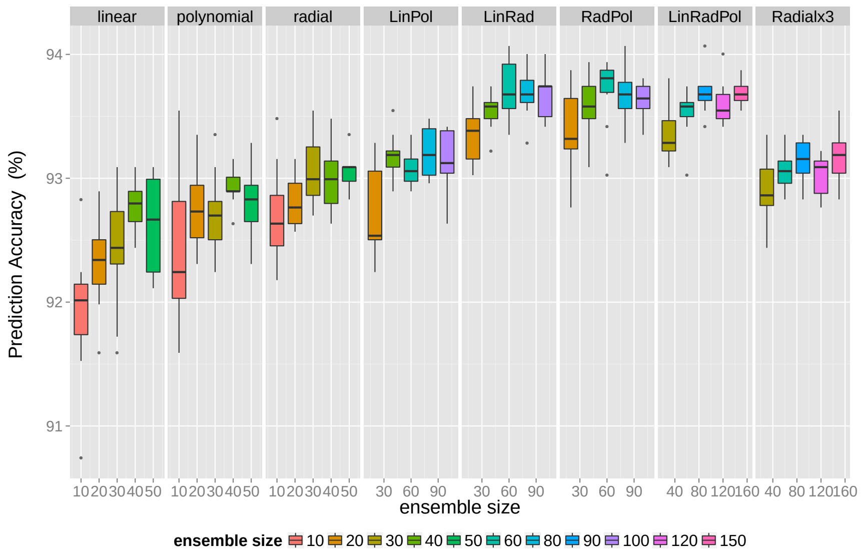

Fig. 1 Accuracy on task Spam with SVM-bagging. Each sub-classifier uses a training sample of only 300 records (roughly 10% of the available training data). Differnt ensembles of pure and mixed kernel type are formed, as indicated in the figure head.

The effectiveness of Bagging is shown in Fig. 1. Ensembles of mixed type (LinRad, RadPol and LinRadPol) are significantly better in this case than ensembles of pure type (Lin, Rad and Pol). Even if we build from the best type (radial) an ensemble ”Radialx3” with the same ensemble size as in LinRadPol, its accuracy is significantly below that of LinRadPol.

The complete results on all datasets are shown in Table 1. It is impressive to note that the ensembles – although each ensemble SVM is only trained on a much

Table 2 Accuracies of the different models in percent. All ensembles were trained on small stratified samples of size 300, while the single SVM was trained on all training data (3000-79000 records). The first line shows the mean test-set accuracy of ten runs with different training samples. The second line (in *italic numbers*) gives the standard deviations. The best result in each row is marked in **bold face**. A result in column *RadPol* is <u>underlined</u> if it is significantly better than the results in the preceeding columns *radial* and *polynom*.

Name	SVM	Bagging					Boosting			
		radial	polynom	RadPol	LinRadPol	Radialx3	radial	polynom	mixed	RadPol
ens.size	1	50	50	100	150	150	50	50	50	100
Spam	93.01	93.05	92.80	<u>93.64</u>	93.70	93.17	93.74	93.94	94.01	**94.10**
		0.14	*0.26*	*0.13*	*0.10*	*0.21*	*0.33*	*0.24*	*0.19*	*0.32*
OptDigit	**97.94**	96.36	96.19	96.33	96.15	96.38	97.69	97.38	97.11	97.69
		0.16	*0.10*	*0.16*	*0.08*	*0.10*	*0.23*	*0.19*	*0.15*	*0.15*
Satlog	**91.05**	87.51	86.49	87.21	86.16	87.76	89.96	88.73	89.69	90.31
		0.28	*0.36*	*0.19*	*0.24*	*0.14*	*0.58*	*0.67*	*0.30*	*0.36*
Adult	84.45	84.84	82.94	84.03	84.50	84.86	84.53	84.15	84.40	<u>**84.92**</u>
		0.07	*0.13*	*0.10*	*0.05*	*0.05*	*0.30*	*0.22*	*0.42*	*0.26*
Acoustic2	**90.70**	90.10	89.85	90.28	90.05	90.26	89.46	89.56	89.28	90.01
		0.13	*0.10*	*0.11*	*0.05*	*0.08*	*0.37*	*0.18*	*0.05*	*0.15*

Table 3 Training times in seconds (Intel Core i73632QM CPU, 64bit, 2.2 GHz, 8GB RAM).

Name	# train records	SVM on all records	Boosting on 300 records, ensemble size 50			
			radial	polynomial	mixed	RadPol
Spam	3067	1.2	11.0	8.0	9.5	19.1
OptDigit	3823	2.2	13.0	11.5	12.4	24.5
Satlog	4435	2.1	12.5	9.2	10.3	21.7
Adult	30162	148.2	60.1	37.7	50.0	97.8
Acoustic2	78823	3866.5	231.3	167.2	199.4	398.6

smaller training set – reach or even surpass the accuracy of the single SVM. For the ensembles we tested all three kernel types, but since radial and polynomial performed usually better than linear, we show only the former pure types. A result in column RadPol is <u>underlined</u>, if it was significantly ($\alpha = 5\%$) better in a t-test comparison to the 'radial' or 'polynom' results of the same method. It has to be noted that RadPol / LinRadPol ensembles have twice / three times as many SVMs as a pure ensemble, requiring much more computation time. If we compare LinRadPol in Table 1 with the column Radialx3, we find that it is sometimes better than even Radialx3 (dataset Spam), but sometimes not (datasets Satlog and Adult).

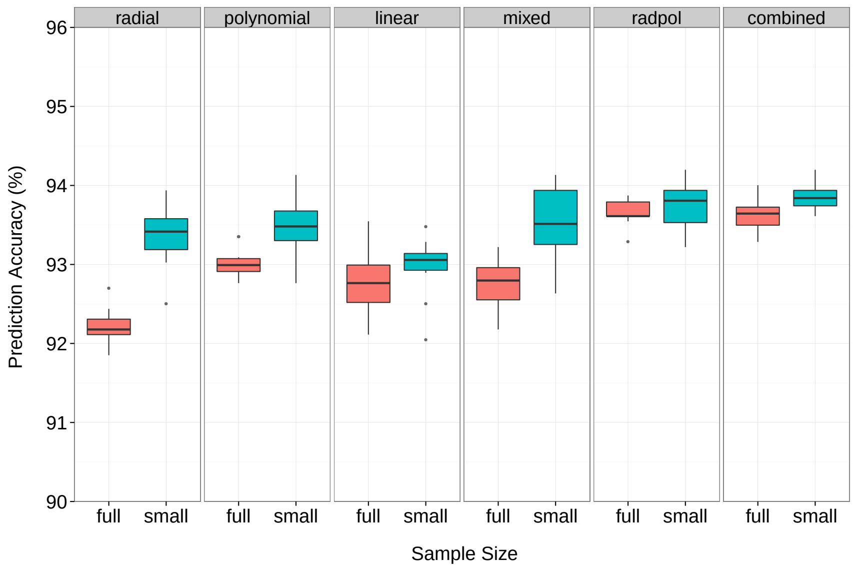

Fig. 2 Test-set accuracy on task Spam with SVM-Boosting. In case of training with the complete data (full: 3036 records), the ensemble suffers from considerable overfitting, while at the same time the ensemble performs better with subsamples of the data (small: 300 records). Mixed ensembles like "radpol" and "combined"=(linear, polynomial, radial) are better than a pure "radial" ensemble.

4.1 Discussion

The most interesting effect is the lower resource consumption for larger datasets. If we compare the training times for the Acoustic2 dataset (Table 3), the RadPol ensemble needs less than $1/10$ and the radial ensemble needs only $1/16$ of the SVM (all data) training time. The price to pay is a somewhat longer prediction time of the ensemble, but SVM prediction is usually fast and the prediction time rises only linearly with ensemble size.

We analyzed the sample size settings used for the ensembles. Here, Bagging usually performs better with larger sample sizes, while Boosting tends to be more effective with smaller sample sizes. The reasons can be interpreted as follows: while for Bagging larger sample sizes lead to better predictions for the single SVM in the ensemble, consequently the ensemble performs better and is more robust. The opposite could be observed for boosting. In boosting, a sequential procedure is initiated, where the decisions in initial iterations directly influence the later behaviour of the algorithm and the underlying prediction models. In our case the ensemble tends to focus on initial wrong predicted patterns, whereas the generalization of the remaining patterns is lost. Thus, the underlying problem can be seen as a certain kind of overfitting. A solution to this can now be given by reducing the sample sizes for the single SVM learners. By conducting a repeated resampling of the data, the overfitting for some individual patterns is avoided. The single learners trained with few patterns seem to be weak at first, but the whole ensemble achieves a better generalization performance by the right combination of several weak learners.

A nice side effect of ensemble learning with SVM is that the aggregation of multiple SVM learners leads to a decreased impact of hyperparameter settings as C or γ. Usually these parameters are very sensitive and have a high influence on the prediction accuracy. The ensemble approach seems to weaken this influence by its subtle combination of the single learners.

Meyer and Bischl [11] did research on parallel approaches for SVMs with Bagging and cascade SVM. We can support their main result that Bagging is often effective in producing good accuracy within drastically reduced time. They had ensembles of size 9, having each $1/9$ of the training data. We found that even lower sample sizes can be effective. In addition we found that AdaBoost gives **only** better results on SVM ensembles, if the training size is small enough.

Wang et al. [13] considered a similar approach to ours in that they proposed a modified AdaBoost algorithm with reduced sample size for each classifier of the ensemble. However, they considered only a slight reduction (80% of all training data) which does not 'weaken' the SVMs sufficiently, and they observe a strong degredation when the ensemble size is increased. We initially observed a similar behavior that the ensemble was less productive than a single SVM when the training sample was too big (see Fig. 2).

5 Conclusion

We performed an analysis of ensemble methods consisting of SVM learners. In our study we compared two ensemble approaches based on Bagging and Boosting. In an experimental study we observed comparable performances of the ensemble learners, with viable runtimes for the ensemble. This could be achieved by incorporating random resampling strategies, considering only parts of the training data for the ensemble learners.

In part of our experiments (Spam/Bagging and Adult/Boosting) the ensemble learners performed significantly better when different kernel types were combined. This is due to the fact that multiple kernel types inside the ensemble are beneficial for diversification, partially supporting our hypothesis H-1. In the other cases the combined ensembles did as well as the pure radial ensembles.

The ensemble performed well enough to compete with one single SVM learner trained on the full data, which corresponds with hypothesis H-3. Due to the bad scalability of SVM, such ensemble approaches are especially interesting for large datasets, where the runtime of a single SVM training becomes prohibitively long. Here, the approaches based on Bagging and Boosting are possible solutions. The most apparent advantage of Bagging is probably its nice potential for paralellization. Boosting could stress its good theoretical properties, but it was necessary to use small training sample sizes for making each SVM in the ensemble weak, as it was advocated by hypothesis H-2.

Acknowledgements This work has been partially supported by the Bundesministerium für Bildung und Forschung (BMBF) under the grant SOMA (AiF FKZ 17N1009) and by the Cologne University of Applied Sciences under the research focus grant COSA.

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

References

1. L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
2. B. Caputo, K. Sim, F. Furesjo, and A. Smola. Appearance-based object recognition using SVMs: Which kernel should I use? *Proc of NIPS workshop on Statistical methods for computational experiments in visual processing and computer vision, Whistler*, 2002.
3. C. Chang and C. Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.
4. E. Y. Chang, K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, and H. Cui. PSVM: Parallelizing support vector machines on distributed computers. *Advances in Neural Information Processing Systems*, 20:16, 2007.
5. C. Cortes and V. Vapnik. Support vector machine. *Machine learning*, 20(3):273–297, 1995.
6. K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2002.
7. N. Cristianini and J. Shawe-Taylor. Support vector machines, 2000.
8. E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, and A. Weingessel. Misc functions of the department of statistics (e1071), TU Wien. *R package*, pages 1–5, 2008.
9. Y. Freund and R. E. Schapire. 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, pages 23–37, London, UK, 1995.
10. J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, 209:415–446, 1909.
11. O. Meyer, B. Bischl, and C. Weihs. Support vector machines on large data sets: Simple parallel approaches. In M. Spiliopoulou et al., editors, *Data Analysis, Machine Learning and Knowledge Discovery*. Springer, 2013.
12. B. Schölkopf and A. Smola. *Learning with kernels: support vector machines, regularization, optimization and beyond.* MIT Press, 2002.
13. S. Wang, A. Mathew, Y. Chen, L. Xi, L. Ma, and J. Lee. Empirical analysis of support vector machine ensemble classifiers. *Expert Systems with Applications*, 36(3):6466–6476, 2009.
14. J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings of the 7th European symposium on artificial neural networks (ESANN)*, volume 99, pages 61–72, 1999.
15. J. Wickramaratna, S. B. Holden, and B. Buxton. Performance degradation in boosting. In J. Kittler and F. Roli, editors, *Proceedings of the 2nd International Workshop on Multiple Classifier Systems*, number 2096 in LNCS, pages 11–21, Cambridge, UK, 2001.
16. H. Yu, J. Yang, J. Han, and X. Li. Making SVMs scalable to large data sets using hierarchical cluster indexing. *Data Mining and Knowledge Discovery*, 11(3):295–321, 2005.