Resilient Approximation of Kernel Classifiers

Thorsten Suttorp and Christian Igel

Institut für Neuroinformatik
Ruhr-Universität Bochum, 44780 Bochum, Germany
{thorsten.suttorp, christian.igel}@neuroinformatik.rub.de

Abstract. Trained support vector machines (SVMs) have a slow run-time classification speed if the classification problem is noisy and the sample data set is large. Approximating the SVM by a more sparse function has been proposed to solve to this problem. In this study, different variants of approximation algorithms are empirically compared. It is shown that gradient descent using the improved Rprop algorithm increases the robustness of the method compared to fixed-point iteration. Three different heuristics for selecting the support vectors to be used in the construction of the sparse approximation are proposed. It turns out that none is superior to random selection. The effect of a finishing gradient descent on all parameters of the sparse approximation is studied.

1 Introduction

Support vector machines (SVMs) show excellent classification performance across a wide range of applications. Unfortunately, the good classification results often come along with extremely slow execution times, which scale linearly with the number of support vectors. As shown in [1], with probability tending to 1 with increasing training sample size, the fraction of training patterns that become support vectors is bounded from below by the Bayes optimal classification rate. That is, for noisy data the run-time complexity of an SVM increases linearly with the size of the training set. But in many real-world applications classifiers have to meet strict real-time constraints. For this reason SVMs—although showing superior classification performance—are frequently not considered. One way to address this problem is to first train an SVM and then to approximate the obtained solution by a more sparse kernel classifier [2–6]. The new solution is a weighted combination of a set of vectors mapped to the feature space plus a bias parameter. The sparse classifier is build incrementally by adding a (support) vector to the set of vectors, adjusting its position, and recomputing all weighting coefficients. In this study, we empirically investigate variants of this algorithm. First, we replace the fixed-point iteration suggested in previous work for placing the new vectors by the improved Rprop algorithm [7], which is an efficient and robust first order gradient method. Second, we propose different heuristics for choosing the new vectors in the incremental procedure and investigate how these techniques influence the final solution. Third, we look at the performance improvements achieved by a computationally expensive finishing optimization of all parameters of the final sparse solution. In our experiments, we monitor the correlation between the distance measure that serves as the optimization criterion during approximation and the accuracy of the resulting sparse solution on test data. Although a high correlation is assumed in the SVM approximation methods, to the best of our knowledge this has never been validated systematically.

In the next section, we briefly review the basic principles for approximating SVMs. In Sect. 3 different variants are proposed. The experiments for evaluating these approximation algorithms are presented in Sect. 4. Finally, the results are discussed.

2 Background

We first concisely describe soft margin SVMs and then the approximation technique proposed in [3–6].

2.1 Support Vector Machines

We consider L_1-norm soft margin SVMs for binary classification [8]. Let (x_i, y_i), $1 \leq i \leq \ell$, be consistent training examples, where $y_i \in \{-1, 1\}$ is the label associated with input pattern $x_i \in \mathcal{X}$. Support vector machines map input patterns to a feature space $\mathcal{F}$, in which the transformed data is linearly separated. The transformation $\phi : \mathcal{X} \to \mathcal{F}$ is implicitly done by a positive semi-definite kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ computing a scalar product in the feature space $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$. Patterns are classified by the sign of a function f of the form

$$f(x) = \langle w, \phi(x) \rangle + b = \sum_{i=1}^{\ell} \alpha_i k(x_i, x) + b \enspace . \tag{1}$$

The coefficients $\alpha_i \in \mathbb{R}$ defining the weight vector $w = \sum_{i=1}^{\ell} \alpha_i \phi(x_i)$ and b are determined by solving the quadratic optimization problem

$$\min_{w,b} \quad H[f] = \sum_{i=1}^{\ell} [1 - y_i f(x_i)]_+ + \frac{1}{2C} \|w\|^2 \enspace , \tag{2}$$

where $[z]_+ = 0$ if $z < 0$ and $[z]_+ = z$ otherwise. The first part penalizes patterns that are not classified correctly with a particular margin (i.e., distance from the separating hyperplane in $\mathcal{F}$). The second part regularizes the solution, in the sense that minimizing the norm of the weight vector corresponds to minimizing the norm of the function $\Psi(x) = \langle w, \phi(x) \rangle$ in $\mathcal{F}$. The regularization parameter $C \in \mathbb{R}^+$ controls the trade-off between the two objectives. The x_i with $\alpha_i \neq 0$ are called support vectors (SVs), their number is denoted with #SV. For solving the dual quadratic optimization problem we use a sequential minimal optimization approach based on second order information as proposed in [9].

2.2 Approximation of SVMs

For many practical applications the calculation of (1) obtained by SVM learning is computationally too expensive. This problem is tackled in [3] by approximating $\Psi(\cdot) = \sum_{i=1}^{\ell} \alpha_i k(x_i, \cdot)$ by a function $\Psi'(\cdot) = \sum_{i=1}^{L} \beta_i k(z_i, \cdot)$ with $L \ll \#\mathrm{SV}$ and $\beta_i \in \mathbb{R}$. The approximation technique proposed there is based on minimizing the distance function

$$\rho^2 := \|\Psi - \Psi'\|_{\mathcal{F}}^2 \ . \tag{3}$$

The reduced set $\{z_1, \dots, z_L\}$ of approximating vectors (AVs) can be constructed by iteratively adding single vectors to a given solution. The following algorithm implements this idea. The different steps are discussed below.

Algorithm 1: Approximation of SVMs

```
1 initialize $\Psi_1 := \Psi$
2 for $i = 1, \dots, L$ do
3     determine $z_i$ by minimizing $\rho^2(z_i) = \|\Psi_i - \beta_i \phi(z_i)\|_{\mathcal{F}}^2$
4     calculate optimal $\beta_1, \dots, \beta_i$
5     $\Psi_{i+1} \leftarrow \Psi - \sum_{j=1}^{i} \beta_j \phi(z_j)$
6 adjust $z_1, \dots, z_L$ and $\beta_1, \dots, \beta_L$ by global gradient descent
7 determine optimal offset $b'$
```

In [5] the authors note that instead of minimizing $\rho^2(z)$ it is possible to directly minimize the distance between Ψ and its orthogonal projection onto span$(\phi(z))$, which is given by $\left\| \frac{(\Psi \cdot \phi(z))}{\phi(z) \cdot \phi(z)} \phi(z) - \Psi \right\|_{\mathcal{F}}^2 = \|\Psi\|_{\mathcal{F}}^2 - \frac{(\Psi \cdot \phi(z))^2}{\phi(z) \cdot \phi(z)}$, and it therefore suffices to minimize $-\frac{(\Psi \cdot \phi(z))^2}{\phi(z) \cdot \phi(z)}$. This expression reduces for kernels with $k(z, z) = 1$ for all z (e.g. Gaussian kernels) to

$$E(z) := -(\Psi \cdot \phi(z))^2 = -\left(\sum_{i=1}^{\ell} \alpha_i k(x_i, z) \right)^2 \ . \tag{4}$$

Minimization of this distance measure can be realized using gradient methods or fixed-point iteration [3–5]. For all of these techniques starting points for the optimization have to be selected.

Fixed-Point Iteration. Determining a single approximation vector z can be performed by means of fixed-point iteration as proposed in [4]. This method is based on the observation that for an extremum of $(\Psi \cdot \phi(z))^2$ the condition $\nabla_z (\Psi \cdot \phi(z))^2 = 0$ has to be fulfilled, which implies $\sum_{i=1}^{\ell} \alpha_i \nabla_z k(x_i, z) = 0$. For kernels with $k(x_i, z) = k(\|x_i - z\|^2)$, such as Gaussian kernels, this reduces to $\sum_{i=1}^{\ell} \alpha_i k'(\|x_i - z\|^2)(x_i - z) = 0$, which is equivalent to

$$z = \frac{\sum_{i=1}^{\ell} \alpha_i k'(\|x_i - z\|^2) x_i}{\sum_{i=1}^{\ell} \alpha_i k'(\|x_i - z\|^2)} \ . \tag{5}$$

Based on this equation, the vector z is computed iteratively. For Gaussian kernels $k(x_i, z) = \exp(-\gamma\|x_i - z\|^2)$ with $\gamma \in \mathbb{R}^+$ the iteration step $t + 1$ reads

$$z^{(t+1)} = \frac{\sum_{i=1}^{\ell} \alpha_i \exp(-\gamma\|x_i - z^{(t)}\|^2) x_i}{\sum_{i=1}^{\ell} \alpha_i \exp(-\gamma\|x_i - z^{(t)}\|^2)} \ . \tag{6}$$

Determining Optimal Coefficients. In each iteration of the approximation algorithm the coefficients β_i have to be determined anew. The optimal coefficients $\beta = (\beta_1, \dots, \beta_L)$ for approximating $\Psi = \sum_{i=1}^{\ell} \alpha_i k(x_i, \cdot)$ by $\Psi' = \sum_{i=1}^{L} \beta_i k(z_i, \cdot)$ for linear independent $\phi(z_1), \dots, \phi(z_L)$ can be computed as $\beta = (K^z)^{-1} K^{zx} \alpha$, where $K^z_{ij} := (\phi(z_i) \cdot \phi(z_j))$ and $K^{zx}_{ij} := (\phi(z_i) \cdot \phi(x_j))$, see [4].

Global Gradient Descent. For further minimizing the distance function ρ^2 gradient descent can be applied to all parameters of the sparse solution [5]. The derivative of ρ^2 with respect to a component of an AV is given by

$$\frac{\partial \rho^2}{\partial (z_i)_k} = 4\gamma \left[\sum_{j=1}^{L} \beta_j \beta_i K^{zz}_{ij} ((z_j)_k - (z_i)_k) - \sum_{j=1}^{\ell} \alpha_j \beta_i K^{zx}_{ij} ((x_j)_k - (z_i)_k) \right] \tag{7}$$

and the derivative with respect to a coefficient β_i by

$$\frac{\partial \rho^2}{\partial \beta_i} = 2 \sum_{j=1}^{L} \beta_l K^{zz}_{ij} - 2 \sum_{j=1}^{\ell} \alpha_j K^{zx}_{ij} \ . \tag{8}$$

Determining Optimal Offset. The offset b used in the original SVM $f(x) = \Psi(x) + b$ is not necessarily optimal for the approximation $f'(x) = \Psi'(x) + b'$. We consider the differences of $f(x)$ and $\Psi'(x)$: $b'(x) = \sum_{i=1}^{\ell} \alpha_i k(x_i, x) + b - \sum_{i=1}^{L} \beta_i k(z_i, x)$ for all SVs of the original SVM and compute their mean

$$b' = \frac{1}{\#\text{SV}} \sum_{x \in \{x_i | 1 \le i \le \ell \wedge \alpha_i \ne 0\}} b'(x) \ . \tag{9}$$

3 Resilient Approximation of SVMs

In order to increase the performance of the SVM approximation algorithm, we consider new heuristics for choosing the starting points for determining the vectors in the reduced set. Further, we replace the fixed-point iteration by a more robust gradient descent technique.

3.1 Techniques for Choosing Initial Vectors

Constructing a reduced set of AVs relies on the successive addition of single vectors. The choice of initial vectors for gradient descent or fixed-point iteration is crucial for the quality of the final solution. Here, three different techniques for choosing initial vectors from the set of SVs of the original SVM are proposed. The techniques *random selection* and *kernel-clustering* ensure that the fraction of positive and negative SVs of the original SVM (i.e., x_i with $y_i = 1$ and $y_i = -1$, respectively) equals the fraction of positive and negative SVs chosen for the incremental update of the approximation. Let n_{pos} denote the number of positive SVs and n_{neg} the number of negative SVs of the original SVM. Then, the number of initial vectors to be chosen from the positive SVs is given by $L_{\text{pos}} := \max\{1, \lfloor \frac{n_{\text{pos}}}{\#\text{SV}} \cdot L \rfloor\}$ and the number from the negative ones by $L_{\text{neg}} := L - L_{\text{pos}}$.

Random Selection. The starting points are selected uniformly at random from the original SVs. No particular assumptions about the order of drawing are made.

α-proportional Selection. The α-proportional selection has its origin in stochastic universal sampling, which is known from evolutionary algorithms [10]. A weighted roulette wheel containing all original SVs is simulated. The slots are sized according to the corresponding $|\alpha_i|$, and L equally spaced markers are placed along the outside of the wheel. The wheel is spun once and the slots that are hit by the markers define the L initial vectors. This heuristic is based on the idea that vectors with big $|\alpha_i|$ are more important than those with small ones.

Kernel-Clustering. Approximated SVMs realize qualitatively different solutions compared to their original SVMs (see below). The AVs tend to lie rather in the center of the training examples than on the boundaries. This inspires us to choose initial vectors for incrementally updating the approximation by the means of clustering.

The well-known K-means-algorithm generates K initial centers at the beginning and then iterates the following procedure until convergence: All vectors are assigned to their closest center, and the new center of each cluster is set to the mean of the vectors assigned to it. In this process it can happen that one cluster remains empty during the iterations. There are some techniques for dealing with this situation. We choose the vector that has the greatest distance from its cluster center to start a new cluster.

The K-means-algorithm can be transferred from clustering data in input space to clustering data in kernel-induced feature space [11]. The kernel function determines the distance of the vectors to the cluster centers. The preimage of the centers cannot be calculated in general. Therefore, the algorithm finally provides pseudo centers, which are given by the input vectors that are closest to the real centers in the feature space. For approximating an SVM, positive and negative SVs are clustered independently into the predetermined number of clusters (L_{pos} and L_{neg}). In [12] the efficient clustering of a large sample set is considered.

3.2 Resilient Minimization of the Distance Function

Although fixed-point iteration has been favored, the optimization of $E(z)$ can also be done using a general gradient descent algorithm [2]. The partial derivatives of $E(z)$ are given by

$$\frac{\partial E(z)}{\partial z_k} = -2 \sum_{i=1}^{\ell} \alpha_i k(x_i, z) \cdot \sum_{i=1}^{\ell} \alpha_i \frac{\partial k(x_i, z)}{\partial z_k} \quad . \tag{10}$$

We employ the efficient Rprop (resilient backpropagation) algorithm for gradient-based optimization [7,13]. It considers only the signs of the partial derivatives of the error function E to be optimized and not their amount. In each iteration t of Rprop, each objective parameter $z_k^{(t)}$ is increased or decreased depending on whether the sign of the partial derivative $\partial E(z^{(t)})/\partial z_k^{(t)}$ of the objective function with respect to the parameter is positive or negative. The amount of the update is equal to the adaptive individual step size $\Delta_k^{(t)}$, that is, we have

$$z_k^{(t+1)} = z_k^{(t)} - \text{sign}\left(\partial E(z^{(t)})/\partial z_k^{(t)}\right) \cdot \Delta_k^{(t)} \quad . \tag{11}$$

Prior to this update, the step size is adjusted based on changes of sign of the partial derivative in consecutive iterations. If the partial derivative changes its sign, indicating that a local minimum has been overstepped, then the step size is multiplicatively decreased; otherwise, it is increased: if $\partial E(z^{(t-1)})/\partial z_k^{(t-1)} \cdot \partial E(z^{(t)})/\partial z_k^{(t)}$ is positive then $\Delta_k^{(t)} = \eta^+ \Delta_k^{(t-1)}$, if the expression is negative then $\Delta_k^{(t)} = \eta^- \Delta_k^{(t-1)}$, where $\eta^+ > 1$ and $\eta^- \in]0,1[$. The iRprop$^+$ algorithm used in this study implements weight backtracking. It partially retracts "unfavorable" previous steps. Whether a parameter change was "unfavorable" is decided based on the evolution of the partial derivatives and the overall error [7]. The standard parameters $\eta^+ = 1.2$ and $\eta^- = 0.5$ reliably give good results and the initial step sizes Δ_k can be chosen small, so that no parameter-tuning is required when applying resilient backpropagation.

4 Experiments

Experimental Setup. We applied the algorithms to the banana data set [14], which is ideal for visualizing the sparse SVM solutions. Optimal parameters (C, γ) for SVM learning were determined by grid search and 5-fold cross-validation on the training data. Second, we used the spam-database data set available from UCI repository [15] having 1,813 positive examples (spam) and 2,788 negative ones. We transformed every feature to zero mean and unit variance. Third, we considered the MNIST handwritten digit database [16], which was split into two classes containing the digits {0,1,2,3,4} and {5,6,7,8,9}, respectively. Forth, we used the connect-4 opening database containing 67,557 game states [15]. For binary classification the "draw" examples were removed resulting in 61,108 data points. The data were split roughly into two halves making up training and test data. Parameters (C, γ) for the problems spam-database, MNIST and connect-4 were taken from [9].

For each benchmark problem, an SVM was trained and then a predefined number of approximation trials was conducted for different numbers of AVs. All three techniques for selecting initial vectors in combination with iRprop$^+$ and fixed-point iteration were considered. A final gradient descent was performed for the two “small” problems banana and spam-database.

Results. We compared different SVM approximation techniques, which differ in the method used for minimizing the distance function and the method for choosing the initial vectors. Surprisingly, all techniques for selecting initial vectors in combination with iRprop$^+$ showed almost the same behavior. Because no significant difference could be observed we confine ourselves to discussing the results with random selection only. Fixed-point iteration performed badly on spam-database, MNIST, and connect-4 in that way that the algorithm was not able to reach the target number of approximation vectors without getting stuck. The repeated choice of different initial vectors did not help.

The 2-dimensional banana data set provided insight into the characteristics of the resulting SVM approximation. Compared to the original SVM qualitatively different solutions (see Fig. 1) were realized. The vectors of the sparse solution lay rather in the center of the training examples than on the boundaries.

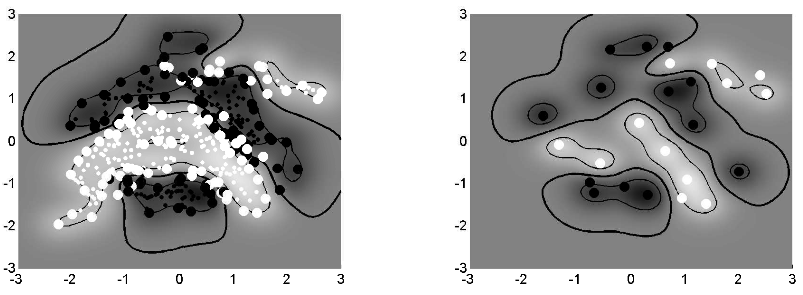

Fig. 1. Approximation of an SVM on the banana data set. The shading codes for the answer of the classifier. The colors white and black correspond to positive and negative examples, respectively. The big points mark the SVs and AVs, respectively. Left image: all training data and the 160 support vectors of the original SVM. Right image: typical approximated SVM with 25 AVs.

The results of the SVM approximation are depicted in Figs. 2-5. In the left and middle pictures the number of AVs is plotted against ρ^2 and the accuracy on the test data, respectively. In addition to the median selected quantiles are given. The fraction of the number of AVs to the number of SVs of the original SVM is reported in brackets. The horizontal line in the middle plots gives the accuracy of the original SVM on the test data. The plots on the right show ρ^2 against the accuracy on the training data. The clusters that can be recognized correspond to solutions with the same number of AVs.

On the banana data set sparse solutions with 20 AVs (e.g., 1/8 of the original amount of SVs) were obtained that performed better on the test data than the original SVM. On spam-database, MNIST and connect-4 data the performance of the original SVM was not fully achieved, but only a small fraction of the number of original SVs led to competitive results (Figs. 3-5).

The final gradient descent on all parameters, which was only tested on banana and spam-database, improved the quality of the approximation. Further, the gradient descent decreased the variance of the solutions in both objectives, distance and accuracy on the test data. As expected, the distance measure ρ^2 was clearly correlated to the accuracy of the approximated SVM on the test data.

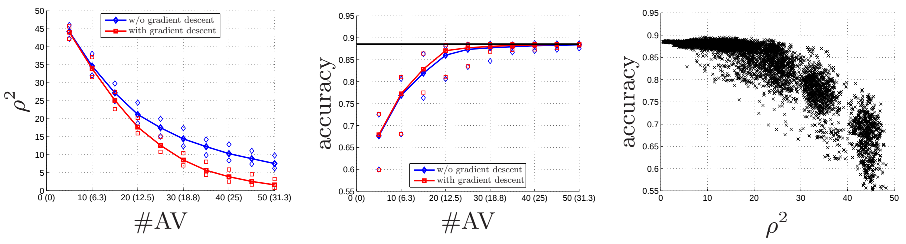

Fig. 2. Approximation results on banana (100 trials); median, 10% and 90% quantiles are given. The fraction of AVs to SVs is reported in brackets.

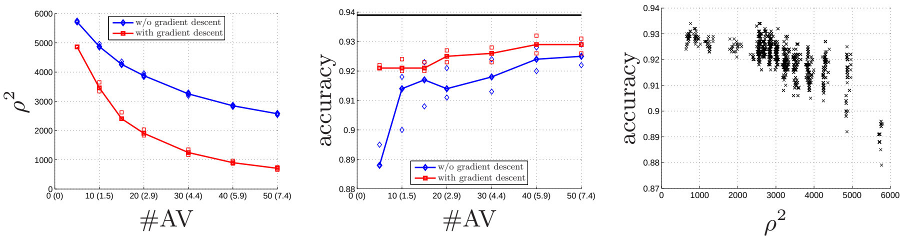

Fig. 3. Approximation results on spam-database (25 trials); median, 10% and 90% quantiles are given.

5 Discussion

The approximation of kernel classifiers allows for the use of SVM training to build classifiers for real-world applications requiring fast decisions. The perfor-

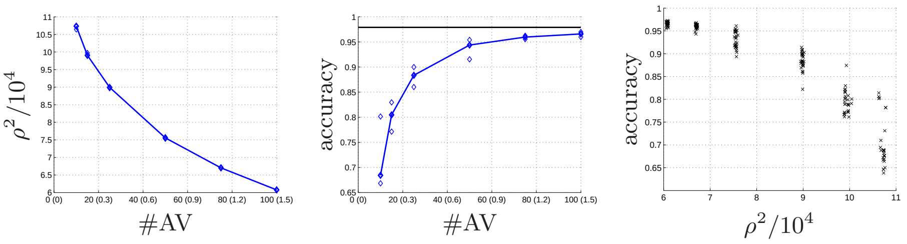

Fig. 4. Approximation results on MNIST (10 trials); median, 20% and 80% quantiles are given.

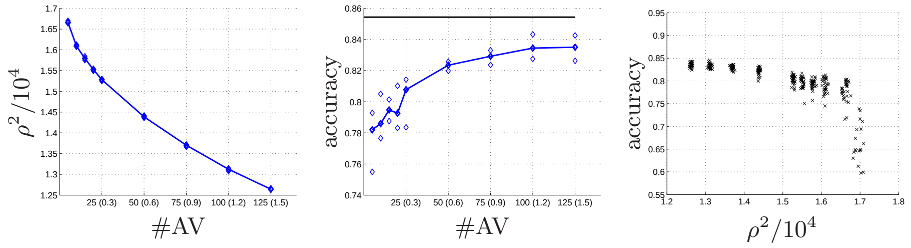

Fig. 5. Approximation results on connect-4 (10 trials); median, 20% and 80% quantiles are given.

mance of the sparse solution may stay behind the original classifier, but the computational complexity is drastically reduced. We found that the SVM approximation provides qualitative different solutions compared to the original SVM. The sparse classifiers are not built on vectors at the decision boundary. The SVM strategy to consider the training patterns at the boundary fails to produce sparse classifiers if the problem is noisy. The newly proposed variant that utilizes the improved Rprop algorithm extends the applicability of SVM approximation compared to fixed-point iteration. The latter technique totally failed on some benchmark problems.

We considered different techniques for choosing initial vectors for the iterations. It turned out that the choice of the selection method for initial vectors did not affect the quality of the final solution. We further analyzed the correlation of the distance function and the classification rates on the test data sets. These two variables were correlated justifying the distance function used for minimization. In accordance to the results in [5], in our experiments a final optimization of all parameters of the sparse models increased the performance. Additionally, we found that the variance of the results produced by the approximation algorithms is reduced.

Thus, we recommend the improved Rprop algorithm for resilient minimization of the distance function in combination with random selection of initial vectors. This algorithm combines simplicity and reliability on all problems considered in this paper. Wherever applicable, a finishing gradient descent on the final solution is recommended to improve the results of the final classifier.

Acknowledgment

Part of the work was funded by L-1 Identity Solutions AG, Bochum, the European Commission, and the State of Northrhine-Westfalia (Germany) as part of the project “Zukunftswettbewerb Ruhrgebiet” (contract no. 0403z010).

References

1. Steinwart, I.: Sparseness of support vector machines. Journal of Machine Learning Research **4** (2003) 1071–1105
2. Burges, C.J.C.: Simplified support vector decision rules. In: Proceedings of the 13th International Conference on Machine Learning (ICML 1996). (1996) 71–77
3. Burges, C.J.C., Schölkopf, B.: Improving the accuracy and speed of support vector machines. In Mozer, M., Jordan, M., Petsche, T., eds.: Advances in Neural Information Processing Systems. Volume 9., Cambridge, MA (1997) 375–381
4. Schölkopf, B., Knirsch, P., Smola, A.J., Burges, C.J.C.: Fast approximation of support vector kernel expansions, and an interpretation of clustering as approximation in feature space. In Levi, P., Ahlers, R.J., May, F., Schanz, M., eds.: DAGM-Symposium, Springer-Verlag (1998) 124–132
5. Schölkopf, B., Mika, S., Burges, C.J.C., Knirsch, P., Müller, K.R., Rätsch, G., Smola, A.J.: Input space versus feature space in kernel-based methods. IEEE Transactions on Neural Networks **10**(5) (1999) 1000–1017
6. Romdhani, S., Torr, P., Schölkopf, B., Blake, A.: Efficient face detection by a cascaded support-vector machine expansion. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **460**(2051) (2004) 3283–3297
7. Igel, C., Hüsken, M.: Empirical evaluation of the improved Rprop learning algorithm. Neurocomputing **50**(C) (2003) 105–123
8. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**(3) (1995) 273–297
9. Glasmachers, T., Igel, C.: Maximum-gain working set selection for support vector machines. Journal of Machine Learning Research **7** (2006) 1437–1466
10. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In Grefenstette, J.J., ed.: Proceedings of the Second International Conference on Genetic Algorithms. (1987) 14–21
11. Girolami, M.: Mercer kernel-based clustering in feature space. IEEE Transactions on Neural Networks **13**(3) (2002) 780–784
12. Zhang, R., Rudnicky, A.I.: A large scale clustering scheme for kernel k-means. In: Proceedings of the International Conference on Pattern Recognition. (2002) 289–292
13. Riedmiller, M.: Advanced supervised learning in multi-layer perceptrons – From backpropagation to adaptive learning algorithms. Computer Standards and Interfaces **16**(5) (1994) 265–278
14. Rätsch, G., Onoda, T., Müller, K.R.: Soft margins for AdaBoost. Machine Learning **42**(3) (2001) 287–320
15. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI repository of machine learning databases (1998)
16. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11) (1998) 2278–2324