Fast Approximation of Support Vector Kernel Expansions, and an Interpretation of Clustering as Approximation in Feature Spaces

Bernhard Schölkopf[1], Phil Knirsch[2], Alex Smola[1], and Chris Burges[2]

[1] GMD FIRST, Rudower Chaussee 5, 12489 Berlin, http://svm.first.gmd.de
[2] Bell Labs, Lucent Technologies, Crawfords Corner Rd., Holmdel NJ 07733, USA

Abstract. Kernel-based learning methods provide their solutions as expansions in terms of a kernel. We consider the problem of reducing the computational complexity of evaluating these expansions by approximating them using fewer terms. As a by-product, we point out a connection between clustering and approximation in reproducing kernel Hilbert spaces generated by a particular class of kernels.

1 Introduction

Recent years have seen a surge in the interest in learning methods based on Mercer kernels, i.e. functions $k(\mathbf{x},\mathbf{y})$ which for all data sets $\{\mathbf{x}_1,\ldots,\mathbf{x}_\ell\} \subset \mathbf{R}^N$ give rise to positive matrices $K_{ij} := k(\mathbf{x}_i,\mathbf{x}_j)$ [10]. Using k instead of a dot product in $\mathbf{R}^N$ corresponds to mapping the data into a possibly high-dimensional space F by a (usually nonlinear) map $\Phi : \mathbf{R}^N \to F$, and taking the dot product there, i.e. [2]

$$k(\mathbf{x},\mathbf{y}) = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})). \tag{1}$$

Any linear algorithm which can be carried out in terms of dot products can be made nonlinear by substituting an a priori chosen kernel. Examples thereof are the Gaussian kernel $k(\mathbf{x},\mathbf{y}) = \exp(-\|\mathbf{x}-\mathbf{y}\|^2/(2\sigma^2))$ and the polynomial kernel $k(\mathbf{x},\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y})^d$ (which corresponds to a dot product in a feature space spanned by *all* products of order d in the original input features [2, 11]). We can think of Φ as a map $\Phi : \mathbf{x} \mapsto k(\mathbf{x},.)$ into a Hilbert space F of functions $\sum_i \alpha_i k(\mathbf{x}_i,.)$ with a dot product satisfying $\langle k(\mathbf{x},.), k(\mathbf{y},.)\rangle_F = k(\mathbf{x},\mathbf{y})$ [11]. By virtue of this property, F is called a *reproducing kernel Hilbert space* (e.g. [10]).

Let us mention two examples of algorithms using Mercer kernels:

Support Vector (SV) classifiers [14] construct a maximum margin hyperplane in F. In input space, this corresponds to a nonlinear decision boundary of the form

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{\ell} \gamma_i k(\mathbf{x},\mathbf{x}_i) + b\right), \tag{2}$$

where the $\mathbf{x}_i$ are the training examples. Those with $\gamma_i \neq 0$ are called *Support*

Vectors; in many applications, most of the γ_i, which are found by solving a quadratic program, turn out to be 0. Excellent classification accuracies in both OCR and object recognition have been obtained using SV machines [11]. A generalization to the case of regression estimation, leading to similar function expansion, exists [14].

Kernel Principal Component Analysis [12] carries out a linear PCA in the feature space F. The extracted features take the nonlinear form

$$f_k(\mathbf{x}) = \sum_{i=1}^{\ell} \gamma_i^k k(\mathbf{x}_i, \mathbf{x}), \tag{3}$$

where, up to a normalization, the γ_i^k are the components of the k-th Eigenvector of the matrix $(k(\mathbf{x}_i, \mathbf{x}_j))_{ij}$.

In both techniques, there is a price that we pay for the nonlinearity: since vectors in F are only given implicitly by expansions in terms of images of training examples under Φ, we need to evaluate the kernel function many times. In the NIST benchmark of 60000 handwritten digits, SV machines are more accurate than any other single classifier [11], however, they are inferior to neural nets in run-time classification speed [5]. In applications where the latter is an issue, it is thus desirable to come up with acceleration methods [4, 5, 9].

The present paper gives an analysis for the case of the Gaussian kernel, which has proven to perform very well in applications [13], and proposes and tests an iteration procedure for computing fast approximations of kernel expansions.

2 The Reduced Set (RS) Method

Given a vector $\Psi \in F$, expanded in images of input patterns $\mathbf{y}_i \in \mathbf{R}^N$,

$$\Psi = \sum_{i=1}^{N_y} \gamma_i \Phi(\mathbf{y}_i), \tag{4}$$

with $\gamma_i \in \mathbf{R}, \mathbf{y}_i \in \mathbf{R}^N$, one can try to approximate it by [4]

$$\Psi' = \sum_{i=1}^{N_z} \beta_i \Phi(\mathbf{z}_i), \tag{5}$$

with $N_z < N_y$, $\beta_i \in \mathbf{R}, \mathbf{z}_i \in \mathbf{R}^N$. To this end, one can minimize [4]

$$\|\Psi - \Psi'\|^2 = \sum_{i,j=1}^{N_y} \gamma_i \gamma_j k(\mathbf{y}_i, \mathbf{y}_j) + \sum_{i,j=1}^{N_z} \beta_i \beta_j k(\mathbf{z}_i, \mathbf{z}_j) - 2 \sum_{i=1}^{N_y} \sum_{j=1}^{N_z} \gamma_i \beta_j k(\mathbf{y}_i, \mathbf{z}_j). \tag{6}$$

The crucial point is that even if Φ is not given explicitely, (6) can be computed (and minimized) in terms of the kernel. Consider first the case $N_z = 1$, i.e.

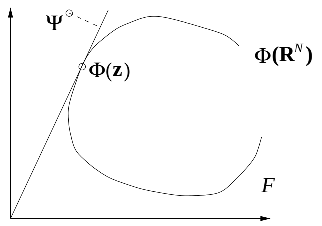

Fig. 1. Given a vector $\Psi \in F$, we try to approximate it by a multiple of a vector $\Phi(\mathbf{z})$ in the image of input space ($\mathbf{R}^N$) under the nonlinear map Φ by finding $\mathbf{z}$ such that the projection distance of Φ onto $\mathrm{Span}(\Phi(\mathbf{z}))$ is minimized.

$\Psi' = \beta\Phi(\mathbf{z})$. We observe that rather than minimizing (6), we can minimize the distance between Ψ and the orthogonal projection of Ψ onto $\mathrm{Span}(\Phi(\mathbf{z}))$ (Fig. 1),

$$\left\| \frac{(\Psi \cdot \Phi(\mathbf{z}))}{(\Phi(\mathbf{z}) \cdot \Phi(\mathbf{z}))}\Phi(\mathbf{z}) - \Psi \right\|^2 = \|\Psi\|^2 - \frac{(\Psi \cdot \Phi(\mathbf{z}))^2}{(\Phi(\mathbf{z}) \cdot \Phi(\mathbf{z}))}. \tag{7}$$

To this end, we maximize

$$\frac{(\Psi \cdot \Phi(\mathbf{z}))^2}{(\Phi(\mathbf{z}) \cdot \Phi(\mathbf{z}))}, \tag{8}$$

which can be expressed in terms of the kernel. The maximization of (8) over $\mathbf{z}$ is preferable to the one of (6) over $\mathbf{z}$ and β, since it comprises a lower-dimensional problem, and since $\mathbf{z}$ and β have different scaling behaviour. Once the maximum of (8) is found, it is extended to the minimum of (6) by setting (cf. (7)) $\beta = (\Psi \cdot \Phi(\mathbf{z}))/(\Phi(\mathbf{z}) \cdot \Phi(\mathbf{z}))$. The function (8) can either be minimized using standard techniques (as in [4]), or, for particular choices of kernels, using fixed-point iteration methods, as shown in the next section.

3 Clustering as Approximation in Feature Space

For kernels which satify $k(\mathbf{z}, \mathbf{z}) = 1$ for all $\mathbf{z} \in \mathbf{R}^N$ (e.g. Gaussian kernels), (8) reduces to

$$(\Psi \cdot \Phi(\mathbf{z}))^2. \tag{9}$$

For the extremum, we have $0 = \nabla_{\mathbf{z}}(\Psi \cdot \Phi(\mathbf{z}))^2 = 2(\Psi \cdot \Phi(\mathbf{z}))\nabla_{\mathbf{z}}(\Psi \cdot \Phi(\mathbf{z}))$. To evaluate the gradient in terms of k, we substitute (4) to get the sufficient condition $0 = \sum_{i=1}^{N_y} \gamma_i \nabla_{\mathbf{z}} k(\mathbf{y}_i, \mathbf{z})$. For $k(\mathbf{x}_i, \mathbf{z}) = k(\|\mathbf{x}_i - \mathbf{z}\|^2)$ (e.g. Gaussians, or $(\|\mathbf{x}_i - \mathbf{z}\|^2 + 1)^c$ for $c = -1, -1/2$), we obtain $0 = \sum_{i=1}^{N_y} \gamma_i k'(\|\mathbf{y}_i - \mathbf{z}\|^2)(\mathbf{y}_i - \mathbf{z})$, leading to

$$\mathbf{z} = \frac{\sum_{i=1}^{N_y} \gamma_i k'(\|\mathbf{y}_i - \mathbf{z}\|^2)\mathbf{y}_i}{\sum_{i=1}^{N_y} \gamma_i k'(\|\mathbf{y}_i - \mathbf{z}\|^2)}. \tag{10}$$

For the Gaussian kernel $k(\mathbf{x}_i, \mathbf{z}) = \exp(-\|\mathbf{x}_i - \mathbf{z}\|^2/(2\sigma^2))$ we thus arrive at

$$\mathbf{z} = \frac{\sum_{i=1}^{N_y} \gamma_i \exp(-\|\mathbf{y}_i - \mathbf{z}\|^2/(2\sigma^2))\mathbf{y}_i}{\sum_{i=1}^{N_y} \gamma_i \exp(-\|\mathbf{y}_i - \mathbf{z}\|^2/(2\sigma^2))}, \tag{11}$$

and devise an iteration

$$\mathbf{z}_{n+1} = \frac{\sum_{i=1}^{N_y} \gamma_i \exp(-\|\mathbf{y}_i - \mathbf{z}_n\|^2/(2\sigma^2))\mathbf{y}_i}{\sum_{i=1}^{N_y} \gamma_i \exp(-\|\mathbf{y}_i - \mathbf{z}_n\|^2/(2\sigma^2))}. \tag{12}$$

The denominator equals $(\Psi \cdot \Phi(\mathbf{z}_n))$ and thus is nonzero in a neighbourhood of the extremum of (9), unless the extremum itself is zero. The latter only occurs if the projection of Ψ on the linear span of $\Phi(\mathbf{R}^N)$ is zero, in which case it is pointless to try to approximate Ψ. Numerical instabilities related to $(\Psi \cdot \Phi(\mathbf{z}))$ being small can thus be approached by restarting the iteration with different starting values.

Interestingly, (12) can be interpreted in the context of clustering (e.g. [3]). It resembles an EM iteration for the determination of the center of a single Gaussian cluster trying to capture as many of the $\mathbf{y}_i$ with positive γ_i as possible, and simultaneously avoids those $\mathbf{y}_i$ with negative γ_i. For SV classifiers, the sign of the γ_i equals the label of the pattern $\mathbf{y}_i$. It is this sign which distinguishes (12) from plain clustering or parametric density estimation. The occurence of negative signs is related to the fact that we are not trying to estimate a parametric density but the *difference* between two densities (modulo normalization constants).

To see this, we define the sets $pos = \{i : \gamma_i > 0\}$ and $neg = \{i : \gamma_i < 0\}$, and the shorthands $p_{pos}(\mathbf{z}) = \sum_{pos} \gamma_i \exp(-\|\mathbf{y}_i - \mathbf{z}\|^2/(2\sigma^2))$ and $p_{neg}(\mathbf{z}) = \sum_{neg} |\gamma_i| \exp(-\|\mathbf{y}_i - \mathbf{z}\|^2/(2\sigma^2))$. The target (9) then reads $(p_{pos}(\mathbf{z}) - p_{neg}(\mathbf{z}))^2$, i.e. we are trying to find a point $\mathbf{z}$ where the difference between the (unnormalized) "probabilities" for the two classes is maximal, and estimate the approximation to (4) by a Gaussian centered at $\mathbf{z}$. Moreover, note that we can rewrite (11) as

$$\mathbf{z} = \frac{p_{pos}(\mathbf{z})}{p_{pos}(\mathbf{z}) - p_{neg}(\mathbf{z})}\mathbf{y}_{pos} + \frac{p_{neg}(\mathbf{z})}{p_{neg}(\mathbf{z}) - p_{pos}(\mathbf{z})}\mathbf{y}_{neg}, \tag{13}$$

where

$$\mathbf{y}_{pos/neg} = \frac{\sum_{pos/neg} \gamma_i \exp(-\|\mathbf{y}_i - \mathbf{z}\|^2/(2\sigma^2))\mathbf{y}_i}{\sum_{pos/neg} \gamma_i \exp(-\|\mathbf{y}_i - \mathbf{z}\|^2/(2\sigma^2))}. \tag{14}$$

4 Multi-Term RBF Approximations

So far, we have dealt with approximations $\Psi' = \beta\Phi(\mathbf{z})$. Now suppose we want to approximate a vector

$$\Psi_1 = \sum_{i=1}^{\ell} \alpha_i \Phi(\mathbf{x}_i) \tag{15}$$

by an expansion of the type (5) with $N_z > 1$. To this end, we iterate [4] by $\Psi_{m+1} := \Psi_m - \beta_m \Phi(\mathbf{z}_m)$. Here, $\mathbf{z}_m$ denotes the $\mathbf{z}$ found for Ψ_m, obtained by iterating (12). To apply (12), Ψ_m needs to be utilized in its representation in terms of mapped input images,

$$\Psi_m = \sum_{i=1}^{\ell} \alpha_i \Phi(\mathbf{x}_i) - \sum_{i=1}^{m-1} \beta_i \Phi(\mathbf{z}_i), \tag{16}$$

i.e. we need to set $N_y = \ell + m - 1$, $(\gamma_1, \ldots, \gamma_{N_y}) = (\alpha_1, \ldots, \alpha_\ell, -\beta_1, \ldots, -\beta_{m-1})$, and $(\mathbf{y}_1, \ldots, \mathbf{y}_{N_y}) = (\mathbf{x}_1, \ldots, \mathbf{x}_\ell, \mathbf{z}_1, \ldots, \mathbf{z}_{m-1})$.

The coefficient $\beta_m = \beta(\mathbf{z}_m)$ could be computed as $(\Psi \cdot \Phi(\mathbf{z}))/(\Phi(\mathbf{z}) \cdot \Phi(\mathbf{z}))$. However, if the $\Phi(\mathbf{z}_1), \ldots, \Phi(\mathbf{z}_m)$ are not orthogonal in F, then the best approximation of Ψ_1 in their Span is *not* obtained by computing orthogonal projections onto each direction. Instead, we need to compute the optimal coefficients $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_m)$ anew in each step. To this end, we evaluate the derivative of the distance in F, $\frac{\partial}{\partial \beta_k} \|\Psi_1 - \sum_{i=1}^m \beta_i \Phi(\mathbf{z}_i)\|^2 = -2\Phi(\mathbf{z}_k)(\Psi_1 - \sum_{i=1}^m \beta_i \Phi(\mathbf{z}_i))$, and set it to 0. Substituting (15) (with $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_\ell)$) and using matrix notation, $K^z_{ij} := (\Phi(\mathbf{z}_i) \cdot \Phi(\mathbf{z}_j))$ and $K^{zx}_{ij} := (\Phi(\mathbf{z}_i) \cdot \Phi(\mathbf{x}_j))$, this leads to $K^{zx}\boldsymbol{\alpha} = K^z \boldsymbol{\beta}$, hence[1]

$$\boldsymbol{\beta} = (K^z)^{-1} K^{zx} \boldsymbol{\alpha}. \tag{17}$$

The iteration is stopped after N_z steps, either specified in advance, or by monitoring when $\|\Psi_{m+1}\|$ (i.e. $\|\Psi_1 - \sum_{i=1}^m \beta_i \Phi(\mathbf{z}_i)\|$) falls below a specified threshold. The solution vector takes the form (5).

5 Approximating Several Vectors Simultaneously

In many applications, we actually would like to approximate more than just one vector in feature space, and we are interested in keeping the total number of approximation vectors small. For instance, in SV digit recognition, one typically trains 10 binary classifiers, and combines the responses. Previous studies have dealt with this by estimating 10 approximations separately [4, 5], which need not be optimal. Even more pronounced is the case of Kernel PCA [12]. There, a network extracting n features computes n projections on vectors in feature space. Approximating each one separately would most likely be wasteful. Approximating the vectors $\Psi^1, \ldots, \Psi^n$ simultaneously results in a network where the computationally expensive first layer is shared by the different sub-networks.

First note that in most cases, already the above algorithm can be used in this respect. Often, the terms in the approximation of one vector will not be orthogonal to the other vectors, and can thus be used, essentially for free, to decrease the discrepancy also for the other vectors, by computing corresponding values of the β_i as in (17).

More principled is the following approch: as a straightforward generalization of (7), we minimize the sum of the projection distances,

$$\sum_{k=1}^n \left\| \frac{(\Psi^k \cdot \Phi(\mathbf{z}))}{(\Phi(\mathbf{z}) \cdot \Phi(\mathbf{z}))} \Phi(\mathbf{z}) - \Psi^k \right\|^2 = \sum_{k=1}^n \left(\|\Psi^k\|^2 - \frac{(\Psi^k \cdot \Phi(\mathbf{z}))^2}{(\Phi(\mathbf{z}) \cdot \Phi(\mathbf{z}))} \right), \tag{18}$$

[1] If the discrepancy Ψ_{m+1} has not yet reached zero, then K^z will be invertible.
If K^z does not have full rank, one can use the Pseudoinverse, or select the solution which has the largest number of zero components.

which amounts to maximizing $\sum_{k=1}^{n}(\Psi^k \cdot \Phi(\mathbf{z}))^2/(\Phi(\mathbf{z}) \cdot \Phi(\mathbf{z}))$. For the Gaussian RBF (radial basis function) kernel, the gradient takes the form $0 = \nabla_{\mathbf{z}}(\Psi^k \cdot \Phi(\mathbf{z}))^2 = 2(\Psi^k \cdot \Phi(\mathbf{z}))\nabla_{\mathbf{z}}(\Psi^k \cdot \Phi(\mathbf{z}))$. For $\Psi^k = \sum_{i=1}^{N_y^k} \gamma_i^k \Phi(\mathbf{y}_i^k)$, we thus obtain

$$\mathbf{z} = \frac{\sum_{k=1}^{n} \sum_{i,j=1}^{N_y^k} \gamma_i^k \exp(-\|\mathbf{y}_i^k - \mathbf{z}\|^2/(2\sigma^2)) \exp(-\|\mathbf{y}_j^k - \mathbf{z}\|^2/(2\sigma^2))\mathbf{y}_j^k}{\sum_{k=1}^{n} \sum_{i,j=1}^{N_y^k} \gamma_i^k \exp(-\|\mathbf{y}_i^k - \mathbf{z}\|^2/(2\sigma^2)) \exp(-\|\mathbf{y}_j^k - \mathbf{z}\|^2/(2\sigma^2))}. \quad (19)$$

6 Experiments

In our experiments on the USPS handwritten digit database (e.g. [11]) of 7291 training patterns and 2007 test patterns (size 16×16), we approximated the SV expansions (2) of ten binary classifiers, each trained to separate one digit from the rest. We used the Gaussian kernel $k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2/128)$ (cf. [13]), and the approximation technique described in Sec. 4.

Table 1 shows classification error results for approximations using varying numbers of RS vectors. For large numbers, the accuracy of the original system can be approached closely. The original SV system had 254 SVs per classifier. To get a speedup by a factor of 10, we have to use the system with 25 RS vectors (RS-25), in which case the classification accuracy drops moderately from 4.4% to 5.1%, which is still competitive with convolutional neural networks on

Table 1. Top: numbers of SVs for the original SV RBF system. Bottom: numbers of test errors for each binary recognizer, and test error rates for 10-class classification. First row: original SV system, with 254 SVs on average; following rows: systems with varying numbers of RS (approximation) vectors (RS-n stands for n vectors) per binary recognizer (optimal threshold re-computed on the training set), computed by iterating one-term approximations, separately for each recognizer. Last two rows: with a subsequent global gradient descent, the results can be further improved (see text).

digit	0	1	2	3	4	5	6	7	8	9	average
#SVs	219	92	316	309	288	340	213	206	304	250	254

	0	1	2	3	4	5	6	7	8	9	10-class
SV-254	16	13	30	17	32	22	11	12	26	17	4.4%
RS-10	26	13	45	49	35	54	22	24	39	24	7.1%
RS-15	17	16	43	50	49	37	14	18	45	35	6.4%
RS-20	27	11	38	30	35	43	12	16	30	25	5.6%
RS-25	21	12	38	32	31	22	12	18	33	28	5.1%
RS-50	18	10	33	28	32	23	12	15	35	27	5.0%
RS-100	14	13	26	22	30	26	11	14	28	23	4.8%
RS-150	13	14	28	32	27	24	12	14	29	26	4.7%
RS-250	12	13	26	26	32	25	11	14	26	24	4.6%
RS2-25	14	14	31	22	30	23	11	14	26	17	4.7%
RS2^{old}-25	16	13	32	19	31	26	11	15	25	18	5.0%

Fig. 2. Complete display of RS-20, with coefficients (Top: recognizer of digit 0,..., bottom: digit 9). Note that positive coefficients (roughly) correspond to positive examples.

that data base [7]. Moreover, we can further improve this result by adding the second phase of the traditional RS algorithm, where a global gradient descent is performed in the space of all $(\mathbf{z}_i, \beta_i)$ [4, 5] (computationally more expensive than the first phase by about two orders of magnitude): this led to an error rate of 4.7%. For the considered kernel, this is slightly better than the traditional RS method, which yielded 5.0% (for polynomial kernels, the latter method led to 4.3% at the same speedup [4]). Figure 2 shows the RS-20 vectors of the 10 binary classifiers. As an aside, note that unlike the approach of [4], our algorithm produces images which do look meaningful (i.e. digit-like).

7 Discussion

We have proposed an approach for approximating kernel expansions, and shown experimentally that it did speed up a Gaussian RBF SV machine. Note that in the Gaussian RBF case, the approximation can never be as good as the original: no approximation with $N_z < N_y$ will lead to $\Psi_m = 0$, since the kernel matrix $K_{ij} = (k(\mathbf{y}_i, \mathbf{y}_j))$ has full rank [8]. Otherwise, *exact* reductions can be obtained, as noted in [11], and, independently, [6]: if $K\boldsymbol{\alpha} = 0$, then we can express one column of K, and hence one vector $\Phi(\mathbf{y}_i)$, as a linear combination of the other ones; e.g. by picking the i^{max} with largest $|\alpha_i|$: $\Phi(\mathbf{y}_{i^{max}}) = -\sum_{i \neq i^{max}} \alpha_i \Phi(\mathbf{y}_i) / \alpha_{i^{max}}$.

As in [5], good results are obtained even though the objective function does not decrease to zero (in our experiments, it was reduced by a factor of 2 to 20 in the first phase, depending on how many RS vectors were computed; the global gradient descent yielded another factor 2 – 3). We conjecture that this is due to the following: in classification, we are not interested in $\|\Psi - \Psi'\|$, but in $\int |\mathrm{sgn}(\sum_{i=1}^{N_y} \gamma_i k(\mathbf{x}, \mathbf{y}_i) + b) - \mathrm{sgn}(\sum_{j=1}^{N_z} \beta_i k(\mathbf{x}, \mathbf{z}_i) + \tilde{b})| dP(\mathbf{x})$, where P is the underlying probability distribution of the patterns (cf. [1]). This is consistent with the fact that the performance of a RS SV classifier can be improved by re-computing an optimal threshold $\tilde{b}$ (which we have done in our experiments).

The previous RS method [4, 5] can be used for any SV kernel; the new one is limited to $k(\mathbf{x}, \mathbf{y}) = k(\|\mathbf{x} - \mathbf{y}\|^2)$. However, it led to slightly better results, interpretable RS images, and an interesting connection between clustering and approximation in feature spaces. It appears intriguing to pursue the question whether this connection could be exploited to form more general types of approximations of SV and kernel PCA expansions by making use of Gaussians of variable widths. Moreover, it should be worthwhile to carry out experiments testing simultaneous approximation approaches as the one proposed in Sec. 5.

Acknowledgement. Part of this work was supported by the DFG (# Ja 379/71), the NeuroColt2 program, the ARC, and carried out at The Australian National University. Thanks to S. Mika, B. Williamson, K. Müller and R. Herbrich for discussions and comments.

References

1. K. Blackmore, R. Williamson, and I. Mareels. Decision region approximation by polynomials or neural networks. *IEEE Trans. Inf. Theory*, 43:903 – 907, 1997.
2. B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on COLT*, pages 144–152, Pittsburgh, PA, July 1992. ACM Press.
3. J. M. Buhmann. Data clustering and learning. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 278–281. MIT Press, 1995.
4. C. J. C. Burges. Simplified support vector decision rules. In L. Saitta, editor, *Proceedings, 13th ICML*, pages 71–77, San Mateo, CA, 1996. Morgan Kaufmann.
5. C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector learning machines. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in NIPS 9*, pages 375–381, Cambridge, MA, 1997. MIT Press.
6. T. Frieß. Personal communication. 1998.
7. Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541 – 551, 1989.
8. C. A. Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22, 1986.
9. E. Osuna and F. Girosi. Reducing run-time complexity in support vector machines. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*. MIT Press, Cambridge, MA, 1998. to appear.
10. S. Saitoh. *Theory of Reproducing Kernels and its Applications*. Longman Scientific & Technical, Harlow, England, 1988.
11. B. Schölkopf. *Support Vector Learning*. R. Oldenbourg Verlag, Munich, 1997.
12. B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299 – 1319, 1998.
13. B. Schölkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE Trans. Sign. Processing*, 45:2758 – 2765, 1997.
14. V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.