Alternatives to Parameter Selection for Kernel Methods

Alberto Muñoz[1], Isaac Martín de Diego[2], and Javier M. Moguerza[2]

[1] University Carlos III de Madrid, c/ Madrid 126, 28903 Getafe, Spain
alberto.munoz@uc3m.es
[2] University Rey Juan Carlos, c/ Tulipán s/n, 28933 Móstoles, Spain
{isaac.martin, javier.moguerza}@urjc.es

Abstract. In this paper we propose alternative methods to parameter selection techniques in order to build a kernel matrix for classification purposes using Support Vector Machines (SVMs). We describe several methods to build a unique kernel matrix from a collection of kernels built using a wide range of values for the unkown parameters. The proposed techniques have been successfully evaluated on a variety of artificial and real data sets. The new methods outperform the best individual kernel under consideration and they can be used as an alternative to the parameter selection problem in kernel methods.

1 Introduction

It is well known that the choice of kernel parameters is often critical for the good performance of Support Vector Machines (SVMs). Nevertheless, to find optimal values in terms of generalization performance for the kernel parameters is an open and hard to solve question. For instance, the effect of RBF kernels parameter within a SVM framework has been studied from a theoretical point of view [5]. Several practical proposals to choose the RBF kernel parameter have been made [14,7,2,13]. However, there is not a simple and unique technique to select the best set of parameters to build a kernel matrix. Our proposal is based on the combination of the different kernel matrices that arise with the use of a range of values for the unkown parameters. Combining kernels provides a solution that minimizes the effect of a bad parameter choice. An intuitive and usual approach to build this combination is to consider linear combinations of the matrices. This is the proposal in [6], which is based on the solution of a semi-definite programming problem to calculate the coefficients of the linear combination. The solution of this kind of optimization problem is computationally very expensive [17]. Recently, a simpler algorithm based on the same ideas for learning a linear combination of kernels has been developed [1]. The main difference is the way in which the weights within the semi-definite programming problem are found.

In this paper we propose several methods to build a kernel matrix from a collection of kernels generated from different values of the unkown parameters

S. Kollias et al. (Eds.): ICANN 2006, Part I, LNCS 4131, pp. 216–225, 2006.

in the kernel function. The functions involved in the proposed methods take advantage of class conditional probabilities and nearest neighbour techniques.

The paper is organized as follows. The general framework for the methods is presented in Section 2. The proposed methods are described in Section 3. The experimental setup and results on artificial and real data sets are described in Section 4. Section 5 concludes.

2 General Framework

As already mentioned, our proposal is based on the generation of a collection of kernel matrices using a wide range of values for the unkown kernel parameters. Once the collection has been built, we will combine the kernels in order to build a unique kernel. We will not focus on the generation step but on the combination step. Different ways to generate parameters, not only for kernels but for many other methods, can be consulted in [15] or in many simulation handbooks.

In order to combine the kernel matrices we make use of the concept of functional combination of kernel matrices. This concept was introduced originally in [10].

Let $K_1, K_2, ...K_M$ be a set of M input kernels defined on a data set X, and denote by K^* the desired output combination. Let y denote the label vector, where for simplicity $y_i \in \{-1, +1\}$ (the extension to the multiclass case is straighforward).

Consider the following (functional) weighted sum:

$$K^* = \sum_{m=1}^{M} W_m \otimes K_m \,, \tag{1}$$

where $W_m = [w_m(x_i, x_j)]$ is a matrix whose elements are nonlinear functions $w_m(x_i, x_j)$, with x_i and x_j data points in the sample, and '$\otimes$' denotes the element by element product between matrices (Hadamard product). We assume that $K_m(x_i, x_j) \in [0, 1] \quad \forall \quad i, j, m$ (otherwise the kernels can be scaled). Notice that if $w_m(x_i, x_j) = \mu_m$, where $\mu_m \,, m = 1, \ldots M$ are constants, then the method reduces to calculate a simple linear combination of matrices:

$$K^* = \sum_{m=1}^{M} \mu_m K_m \,. \tag{2}$$

Several methods have been suggested to learn the coefficients μ_m of the linear combination [1,6]. Thus, the formulation used in these papers is a particular case of the formula we use. For instance, if we take $\mu_m = \frac{1}{M}$, the average of the kernel matrices is obtained.

Regarding our proposals, consider the (i, j) element of the matrix K^* in (1):

$$K^*(x_i, x_j) = \sum_{m=1}^{M} w_m(x_i, x_j) K_m(x_i, x_j) \,. \tag{3}$$

This is the general formula of our approximation. In this way, we will generate a particular weight for each pair of elements under consideration.

An aspect that has to be treated before describing the methods is the fact that the kernel matrix arising from the combination has to be a positive semi-definite matrix. Since this can not be guaranteed in advance, we make use of some of the several solutions that have been proposed to solve this difficulty [12]. For instance, consider the spectral decomposition $K^* = Q\Lambda Q^T$, where Λ is a diagonal matrix containing (in decreasing order) the eigenvalues of K^*, and Q is the matrix of the corresponding eigenvectors. Assume that Λ has at least p positive eigenvalues. We can consider a p-dimensional representation by taking the first p columns of Q: $Q_p \Lambda_p Q_p^T$. We will refer to this technique as 'Positive Eigenvalue Transformation'. A computationally cheaper solution is to consider the definition of a new kernel matrix as K^{*2}. Notice that, in this case, the new kernel matrix is: $Q\Lambda^2 Q^T$. We call this method 'Square Eigenvalue Transformation'. In practice, there seems not to be a universally best method to solve this problem [11].

3 Some Specific Proposals

The next section is devoted to described a common aspect to the methods we will propose: The use of conditional class probabilities in order to build the weights $w_m(x_i, x_j)$ introduced in the previous section.

3.1 Conditional Class Probabilities

Consider the pair (x_i, y_i) and an unlabeled observation x_j. Given the observed value x_j, define $P(y_i|x_j)$ as the probability of x_j being in class y_i. If x_i and x_j belong to the same class this probability should be high. Unfortunately, this probability is unknown and has to be estimated. In our proposals, we will estimate it by:

$$P(y_i|x_j) = \frac{n_{ij}}{n}\,, \tag{4}$$

where n_{ij} is the number of the n-nearest neighbours of x_j belonging to class y_i.

Notice that each kernel induces a different type of neighborhood. In fact, there is an explicit relation between a kernel matrix and a distance matrix. For instance, consider a matrix K of inner products in an Euclidean space $\mathcal{F}$ (a kernel). Then $D^2 = ve^T + ev^T - 2K$ is a matrix of square Euclidean distances in $\mathcal{F}$ [4], where v is a vector made up of the diagonal elements of K. Hence, it is advisable to estimate this probability for each kernel representation, that is, for the kernel K_m we will estimate the conditional probabilities $P_m(y_i|x_j)$ using the induced distances matrix D_m^2. We will need the average of this conditional probabilities over the kernel matrices:

$$\bar{P}(y_i|x_j) = \frac{1}{M}\sum_{m=1}^{M} P_m(y_i|x_j)\,, \tag{5}$$

and

$$\bar{\rho}(x_i, x_j) = \frac{\bar{P}(y_i|x_j) + \bar{P}(y_j|x_i)}{2}\,. \tag{6}$$

To estimate the conditional class probabilities, the appropriate size of the neighbourhood has to be determined. We propose a dynamic and automatic method: given two points x_i and x_j, we look for the first common neighbour. For each data point (x_i and x_j), the size k of the neighbourhood will be determined by the number of neighbours nearer than the common neighbour. To be more specific, let $R(x_i, n) = \{n\text{-}nearest\ neighbours\ of\ x_i\}$, then $k = \text{argmin}_n\{R(x_i, n) \cap R(x_j, n) \neq \emptyset\}$. Obviously, the size k of the neighbourhood depends on the particular pair of points under consideration.

At this point, we have the means to implement some particular proposals of combination methods.

3.2 The 'MaxMin' Method

The 'MaxMin' method (first used in [10]) produces a functional combination of two kernels, namely, the maximum and the minimum of the ordered sequence of kernels, being zero the weight assigned to the rest of the kernels. Consider the ordered sequence:

$$\min_{1\leq m\leq M} K_m(x_i, x_j) = K_{[1]}(x_i, x_j) \leq \ldots \leq K_{[M]}(x_i, x_j) = \max_{1\leq m\leq M} K_m(x_i, x_j)\,,$$

where the subscript $[\cdot]$ denotes the position induced by the order. This method builds each element of K^* using the formula:

$$K^*(x_i, x_j) = \bar{\rho}(x_i, x_j)K_{[M]}(x_i, x_j) + (1 - \bar{\rho}(x_i, x_j))K_{[1]}(x_i, x_j)\,. \tag{7}$$

If x_i and x_j belong to the same class then the conditional class probabilities $\bar{\rho}(x_i, x_j)$ will be high and the method guarantees that $K^*(x_i, x_j)$ will be large. On the other hand, if x_i and x_j belong to different classes the conditional class probabilities $\bar{\rho}(x_i, x_j)$ will be low and the method will produce a value close to the minimum of the kernels. In the following, this method will be refered as **MaxMin**.

For the particular case of $K_1, \ldots, K_M$ being a collection of RBF kernels:

$$K_m(x_i, x_j) = e^{-\|x_i - x_j\|^2/(2*\sigma_m^2)}\,, \quad m = 1, \ldots, M\,, \tag{8}$$

with different σ_m values, then, the minimum and the maximum for each pair of points (x_i, x_j) correspond respectively to the highest and the lowest value of the collection of σ_m parameters.

3.3 The Percentile-in Method

Next, we propose a method whose assignment of positive weights $w_m(x_i, x_j)$ is based on the order induced by the kernels. The method builds each element of K^* using the following formulae:

$$K^*(x_i, x_j) = K_{\lceil \bar{\rho}(x_i, x_j)M \rceil}\,, \tag{9}$$

where the subscript $\lceil \cdot \rceil$ denotes the upper rounding of the argument.

We denote this method by **'Percentile-in'** method [10] . If the class probability $\bar{\rho}(x_i, x_j)$ is high, we can expect a high similarity between x_i and x_j and the method will guarantee a high $K^*(x_i, x_j)$. If the class probability $\bar{\rho}(x_i, x_j)$ is low, $K^*(x_i, x_j)$ will be also low.

3.4 The Percentile-out Method

As in the previous method, the last proposed technique is based on the order induced by the kernels. However, in this case two kernel values are considered. Each element of the K^* matrix is built as follows:

$$K^*(x_i, x_j) = \frac{1}{2}\left(K_{\lceil \bar{P}(y_i|x_j)M \rceil} + K_{\lceil \bar{P}(y_j|x_i)M \rceil}\right) , \tag{10}$$

where the subscript $\lceil \cdot \rceil$ denotes the upper rounding of the argument. We denote this method by **'Percentile-out'** method [10] .

If the conditional class probabilities $\bar{P}(y_i|x_j)$ and $\bar{P}(y_j|x_i)$ are high, we can expect a high similarity between x_i and x_j and both methods will guarantee a high $K^*(x_i, x_j)$. If the conditional class probabilities $\bar{P}(y_i|x_j)$ and $\bar{P}(y_j|x_i)$ are both low, $K^*(x_i, x_j)$ will be also low.

This method can be viewed as a smoothed MaxMin method. As in the MaxMin method, two kernels are considered for each pair of points in the sample. The conditional probabilities are used in order to obtain values not so extreme as the maximum and the minimum. Only if $\bar{P}(y_i|x_j) = \bar{P}(y_j|x_i)$ the Percentile-out method reduces to the Percentile-in method. Otherwise, this method takes into account the difference between the proportion of neighbours of x_j belonging to class y_i and the proportion of neighbours of x_i belonging to class y_j.

4 Experiments

To test the performance of the proposed methods, an SVM (with the upper bound on the dual variables fixed to 1) has been trained on several real data sets using the kernel matrix K^* constructed.

In order to classify a non-labelled data point x, $K^*(x, i)$ has to be evaluated. We calculate two different values for $K^*(x, i)$, the first one assumming x belongs to class $+1$ and the second assumming x belongs to class -1. For each assumption, we compute the distance between x and the SVM hyperplane and assign x to the class corresponding to the largest distance from the hyperplane.

Since our technique is based on the calculation of the nearest neighbours, we have compared the proposed methods with the k-Nearest Neighbour classification (k-NN, using the optimal value $k = l^{\frac{4}{d+4}}$ [16]). We have compared our methods with the linear combination of kernels method (LC) [1]. In order to evaluate the improvement provided by our proposals, we have carried out a Wilcoxon signed-rank test (see for instance [8]). This nonparametric test is used to compare the median of the results for different runs of each method. So, the null hypothesis of the test is that our methods do not improve the existing combination techniques.

4.1 Artificial Data Sets

4.2 Two Areas with Different Scattering Matrices

This data set, shown in Figure 1, is made up of 400 points in $\mathbb{R}^2$. Visually there are two areas of points (80% of the sample is in area A_1 and 20% is in area A_2). Each area A_i corresponds to a circle with radio σ_i. Here $\sigma_1 = 10^{-2}\sigma_2$, with $\sigma_2 = 1$. The first group center is $(0, 1)$ and the second group center is $(1, 1)$. Nevertheless, the areas do not coincide with the classes $\{-1, +1\}$ that are to be learned. Half of the points in each class belongs to aread A_1, and the other half to area A_2. Within each area, the classes are linearly separable. Therefore the only way to built a classifier for this data set is to take into account the area each point belongs to. We use 50% of the data for training and 50% for testing.

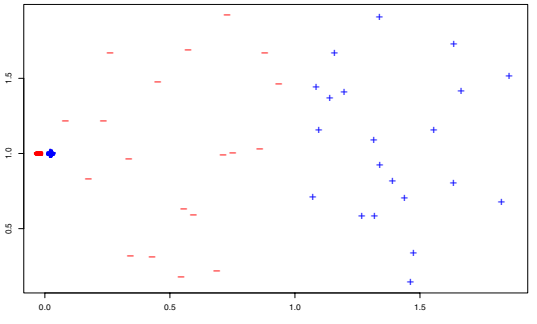

Fig. 1. Two areas with different scattering matrices. The first area center is $(0, 1)$ and the second area center is $(1, 1)$. The areas do not coincide with the classes $\{-1, +1\}$.

Let $\{K_1, \ldots, K_5\}$ be a set of five RBF kernels with parameters σ =0.5, 2.5, 5, 7.5 and 10 respectively. In order to get a positive semi-definite kernel matrix K^*, we use the Square Eigenvalue Transformation technique described in Section 2.

Table 1 shows the performance of our proposals for this data set. The results have been averaged over 10 runs. Given the geometry of the data, it is clear that is not possible to choose a unique best σ for the whole data set. As σ grows, the test error increases for the data contained in area A_1, and decreases within area A_2. The LC method seems to work fairly. Nevertheless, the MaxMin method achieves the best results on classification. Regarding the Wilcoxon signed-rank test for the comparison of our methods with the LC technique, the p-value is smaller than 0.001 for the MaxMin method.

4.3 The Three Spheres Example

The data set contains 300 data points in $\mathbb{R}^4$. We generate three different groups of observations (100 observations per group) corresponding to three spheres in $\mathbb{R}^3$. The center is the same for the three spheres $(0, 0, 0)$ and the radii are

Table 1. Percentage of missclassified data and percentage of support vectors for the two different scattering data set: A_1 stands for the less scaterring group, A_2 stands for the most dispersive one

	Train Error			**Test Error**			**Support Vectors**		
Method	Total	A_1	A_2	Total	A_1	A_2	Total	A_1	A_2
$\mathbf{RBF}_{\sigma=0.5}$	2.1	2.6	0.0	13.5	4.1	51.0	39.6	25.1	97.5
$\mathbf{RBF}_{\sigma=2.5}$	4.8	6.0	0.0	13.5	6.5	41.5	62.2	53.4	97.5
$\mathbf{RBF}_{\sigma=5}$	6.6	8.2	0.0	14.0	10.1	29.5	82.8	79.2	97.0
$\mathbf{RBF}_{\sigma=7.5}$	16.0	19.9	0.5	22.2	22.6	20.5	94.6	94.2	96.0
$\mathbf{RBF}_{\sigma=10}$	30.7	38.2	0.5	37.3	44.1	10.0	94.2	95.4	89.5
MaxMin	0.3	0.4	0.0	4.9	0.9	21.0	27.7	9.6	100.0
Percentile-in	4.2	5.1	0.5	9.0	3.1	32.5	35.9	20.1	99.0
Percentile-out	0.7	0.9	0.0	7.7	1.1	34.0	29.0	11.4	99.5
k**-NN**	14.5	3.5	58.5	15.5	3.5	63.5	—	—	—
LC	1.6	2.0	0.0	8.1	2.5	29.5	46.6	33.2	100.0

different (0.1,0.3, and 1 respectively). The 100 points on the sphere with radio equals to 0.3 belong to class +1, and the other 200 points belong to class −1. Finally a fourth random additional dimension is added to the data set, following a Normal distribution (centered in 0 and with 10^{-2} as standard deviation). We use 50% of the data for training and 50% for testing.

Let $\{K_1, \ldots, K_5\}$ be a set of polynomial kernels, $K(x, z) = (1 + x^T z)^d$, with parameters $d = 1$, 2, 3, 4, 5 respectively. In order to scale the matrices, we use the following normalization [3]: $K(x, z) = K(x, z)/(\sqrt{K(x, x)}\sqrt{K(y, y)})$. We use the Square Eigenvalue Transformation method to solve the problem of building a positive semi-definite matrix. Table 2 shows the performance of the proposed methods when combining these kernel matrices. The results have been averaged over 10 runs.

The MaxMin and Percentile methods show the best overall performance. All our combination methods provide better results than the best polynomial kernel, using usually significantly less support vectors. Regarding the Wilcoxon signed-rank test for the comparison of our methods with the LC technique, the p-values are smaller than 0.001 for all our methods. So the improvement obtained by the use of our proposals is statistically significant. Notice that the results using any of the single kernels are very poor, while the results obtained using any of our combination methods are significatively better. This example also shows that using a linear combination of the kernels may not be a good choice.

4.4 A Real Data Set

In this section we have dealt with a database from the UCI Machine Learning Repository: the Breast Cancer data set [9]. The data set consists of 683 observations with 9 features each. Let $\{K_1, \ldots, K_{12}\}$ be a set of RBF kernels with

Table 2. Percentage of missclassified data, sensitivity (Sens.), specificity (Spec.) and percentage of support vectors for the three spheres data set. Standard deviations in brackets.

Method	Train			Test			Support Vectors
	Error	Sens.	Spec.	Error	Sens.	Spec.	
Polynomial$_{d=1}$	31.8 (2.5)	0.000	1.000	34.9 (2.5)	0.000	1.000	69.5 (5.0)
Polynomial$_{d=2}$	31.8 (2.5)	0.000	1.000	34.9 (2.5)	0.000	1.000	75.7 (7.9)
Polynomial$_{d=3}$	30.6 (1.8)	0.200	0.909	36.1 (1.8)	0.200	0.891	71.7 (5.6)
Polynomial$_{d=4}$	23.7 (7.3)	0.377	0.893	31.7 (7.0)	0.293	0.816	69.5 (4.6)
Polynomial$_{d=5}$	14.7 (2.5)	0.541	0.958	24.1 (7.0)	0.436	0.798	69.5 (4.6)
MaxMin	4.0 (0.8)	0.964	0.958	5.5 (2.5)	0.921	0.958	8.4 (1.2)
Percentile-in	5.5 (1.4)	0.907	0.963	6.9 (3.2)	0.864	0.967	7.6 (1.4)
Percentile-out	4.5 (1.1)	0.941	0.959	6.9 (2.9)	0.886	0.957	8.5 (1.5)
k**-NN**	10.9 (2.4)	0.795	0.934	15.7 (4.2)	0.725	0.904	— (—)
LC	31.8 (2.5)	0.000	1.000	34.9 (2.5)	0.000	1.000	71.5 (3.9)

parameters σ =0.1, 1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 respectively. We use the Positive Eigenvalue Transformation to solve the problem of building a positive semi-definite matrix.

Table 3 shows the performance of the proposed methods when combining all these kernel matrices. Again, the results have been averaged over 10 runs. The MaxMin method, the Percentile-in method, and the Percentile-out method improve the best RBF kernel under consideration (test errors of 2.8% for the three methods vs. 3.1%). The results provided by all the combination methods are not degraded by the inclusion of kernels with a bad generalization performance. Our methods clearly outperform the SVM classifier using an RBF kernel with $\sigma = \sqrt{d}/2$, where d is the data dimension (see [14] for details). Regarding the

Table 3. Percentage of missclassified data, sensitivity (Sens.), specificity (Spec.) and percentage of support vectors for the cancer data using a battery of RBF kernels. Standard deviations in brackets.

Method	Train			Test			Support Vectors
	Error	Sens.	Spec.	Error	Sens.	Spec.	
Best RBF	2.3 (0.3)	0.979	0.976	3.1 (1.6)	0.976	0.966	13.6 (1.3)
Worst RBF	0.0 (0.0)	1.000	1.000	24.7 (2.3)	1.000	0.627	74.0 (2.4)
MaxMin	0.1 (0.1)	0.999	0.998	2.8 (1.6)	0.963	0.975	14.2 (1.5)
Percentile-in	2.0 (0.4)	0.982	0.979	2.8 (2.8)	0.975	0.969	7.8 (0.7)
Percentile-out	0.2 (0.1)	0.999	0.997	2.8 (1.7)	0.964	0.975	19.2 (4.5)
k**-NN**	2.7 (0.5)	0.961	0.980	3.4 (1.5)	0.949	0.974	— (—)
LC	0.0 (0.0)	1.000	1.000	3.2 (1.6)	0.976	0.964	41.5 (4.4)
SVM	0.1 (0.1)	1.000	0.999	4.2 (1.4)	0.989	0.942	49.2 (1.0)

Wilcoxon signed-rank test for the comparison of our methods with the SVM technique, the p-values are smaller than 0.05 for the MaxMin and the Percentile-out methods, and smaller than 0.1 for the Percentile-in method. Again, the improvement obtained by the use of our proposals is statistically significant.

5 Conclusions

In this paper, we have proposed alternative methods to parameter selection techniques within a Kernel Methods framework. The proposed techniques are especially usefull when does not exist an overall and unique best parameter. The suggested kernel combination methods compare favorably to the use of one of the kernels involved in the combination. We have also shown that a linear combination of the kernels may not be a good choice. Further research will focus on the theoretical properties of the methods and extensions. In particular, the methods shown in this paper do not take full advantage of the concept of the functional weighted sum described in (1): we think that there is room for improvement and more sophisticated ways for the calculus of the weights may be designed. The method could be generalized by using more than two kernels, but then, a parameter to control the relative importance of the kernels will be needed [10].

Acknowledgments. This work was partially supported by Spanish grants TIC-2003-05982-C05-05 (MCyT) and PPR-2004-05 (URJC).

References

1. O. Bousquet and D.J.L. Herrmann. On the complexity of learning the kernel matrix. In S. Becker, S. Thurn, and K. Obermayer, editors, *Advances in Neural Information Processing Systems, 15*, pages 415–422. Cambridge, MA: The MIT Press, 2003.
2. O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1/3):131–159, 2002.
3. N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. *On Kernel-Target Alignment*, pages 367–373. Cambridge, MA: MIT Press, 2002.
4. J. C. Gower and P. Legendre. Metric and euclidean properties of dissimilarity coefficients. *Journal of Classification*, 3:5–48, 1986.
5. S.S. Keerthi and C. Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural Computation*, 15:1667–1689, 2003.
6. G. R. G. Lanckriet, N. Cristianini, P. Barlett, L. *El Ghaoui*, and M. I. Jordan. Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research*, 5(Jan):27–72, 2004.
7. J.-H. Lee and C.-J. Lin. Automatic model selection for support vector machines. Technical report, National Taiwan University, 2000.
8. E. L. Lehmann. *NonParametrics: Statistical Methods Based on Ranks*. McGraw-Hill, 1975.
9. O. L. Mangasarian and W. H. Wolberg. Cancer diagnosis via linear programming. *SIAM News*, 23(5):1–18, 1990.

10. J. M. Moguerza, I. Martín de Diego, and A. Muñoz. Improving support vector classificacion via the combination of multiple sources of information. In *Proc. of the IAPR International Workshops SSPR 2004 and SPR 2004, Vol. 3138 of LNCS*, pages 592–600. Berlin: Springer, 2004.
11. E. Pękalska, R. P. W. Duin, S. Günter, and H. Bunke. On not making dissimilarities euclidean. In *Proc. of the IAPR International Workshops SSPR 2004 and SPR 2004, Vol. 3138 of LNCS*, pages 1145–1154. Berlin: Springer, 2004.
12. E. Pękalska, P. Paclík, and R. P. W. Duin. A generalized kernel approach to dissimilarity-based classification. *Journal of Machine Learning Research, Special Issue on Kernel Methods*, 2(12):175–211, 2001.
13. K. Schittkowski. Optimal parameter selection in support vector machines. *Journal of Industrial and Management Optimization*, 1(4):465–476, 2005.
14. B. Schölkopf, S. Mika, C. J.C. Burges, K.-R. Müller P. Knirsch, G. Rätsch, and A. J. Smola. Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 1999.
15. J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
16. B. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, 1986.
17. L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.