Neural Processing Letters **19:** 63–72, 2004.

Minimizing the Cross Validation Error to Mix Kernel Matrices of Heterogeneous Biological Data

KOJI TSUDA[1,2], SHINSUKE UDA[1,3], TAISHIN KIN[1] and KIYOSHI ASAI[1,4]
[1]*AIST Computational Biology Research Center, 2-41-6 Aomi Koto-ku, Tokyo, 135-0064, Japan. e-mail: koji.tsuda@aist.go.jp*
[2]*Max Planck Institute for Biological Cybernetics, Spemannstr. 38, 72076 Tubingen, Germany*
[3]*Department of Computational Intelligence and Systems Science, Tokyo Institute of Technology, Yokohama 2268502, Japan*
[4]*Department of Computational Biology, Graduate School of Frontier Science, University of Tokyo, 5-1-5, Kashiwanoha, Kashiwa 2778562, Japan*

Abstract. In biological data, it is often the case that objects are described in two or more representations. In order to perform classification based on such data, we have to combine them in a certain way. In the context of kernel machines, this task amounts to mix several kernel matrices into one. In this paper, we present two ways to mix kernel matrices, where the mixing weights are optimized to minimize the cross validation error. In bacteria classification and gene function prediction experiments, our methods significantly outperformed single kernel classifiers in most cases.

Key words. bacteria classification, bioinformatics, kernel machines, mixing kernel matrices

1. Introduction

In kernel machines such as support vector machines (SVM) [10], objects are represented as a kernel matrix, where n objects are represented as an $n \times n$ positive semidefinite matrix. Essentially the (i,j) entry of the kernel matrix describes the similarity between ith and jth objects. Due to positive semidefiniteness, the objects can be embedded as n points in an Euclidean space (i.e. the feature space) such that the inner product between two points equals to the corresponding entry of kernel matrix. This property enables us to apply diverse learning methods (e.g. SVM or kernel PCA) without explicitly constructing a feature space [10].

In biological data, it is often the case that objects are described in several heterogeneous representations. For example, the regulatory relationship between genes are represented as a gene network and a gene expression array at the same time [4, 13]. A bacterium has several marker proteins which can be used for classification [15]. When kernel matrices are derived from two different representations, we have two kernel matrices at the same time. In order to use a kernel machine, we have to combine these matrices into one matrix. To this aim, several *mixing* methods have been proposed recently. Pavlidis *et al.* [7] simply took the sum of two matrices obtained from a phylogenetic profile [8] and gene expression. Hanisch *et al.* [4] summed two

distances from a biological network and gene expression. Vert and Kanehisa [13] recently performed kernel CCA to unify two kernel matrices.

In supervised classification problems, we have class labels in training data which may be useful in kernel mixing. The drawback of existing approaches mensioned so far is that they do not use class labels. In this paper, we propose a mixing scheme specialized for supervised classification. For mixing kernel matrices, two kinds of parametric models are adopted, i.e. *linear mixture* and *nonlinear mixture*. The parameters of these models (i.e., mixing weights) are optimized to minimize the *cross validation error*, which is commonly used as an accurate estimate for generalization error (e.g., [2]). In bioinformatics, the support vector machine [10] is often used as a kernel classifier (e.g., [7]), but we utilize the kernel Fisher discriminant analysis [6, 9] because the cross validation error is represented in a closed form. In SVMs, the cross validation error is only approximated by several bounds (e.g., [2]), which may sometimes be inaccurate [3]. We will derive the derivatives of the cross validation error and the mixing weights are optimized in a gradient descent fashion.

We will actually perform experiments in two biological problems: the classification of bacterial proteins and gene function prediction.

Automatic classification of bacteria genera is usually performed with comparing specific proteins contained in the cells. The difference in amino acid sequences is considered as the product of evolutionary changes, but most proteins are not suitable for measuring the similarity between bacteria, because they are subject to abrupt and irregular changes in the evolution process. It is known that the two proteins gyrB and rpoD are especially stable and known to be good markers for measuring similarity [15]. The classification experiments based on one marker protein are done so far e.g., in [14, 15]. What we aim here is to improve classification performance by mixing the kernel matrices of two marker proteins.

The gene function prediction is another important issue in biology [7]. The problem is to find the function of proteins based on several kinds of data. Whereas amino acid sequences offer important features for function prediction, other data such as phylogenetic profiles and gene expression are also found to be useful [7]. In the experiments shown later, we will combine the kernel matrices derived from sequences and phylogenetic profiles. Here the gene expression was not used because it is too noisy and did not help classification performance in our problems.

2. Mixing Kernel Matrices

We consider the following supervised two class classification problem in the domain $\mathcal{X}$. Let $\{x_i, y_i\}_{i=1}^n$, $x_i \in \mathcal{X}, y_i \in \{-1, 1\}$ be the set of labeled training samples, and $x_{n+1}, \ldots, x_{n+m}$ are the set of unlabeled test samples. Our task is to predict the class labels $y_{n+1}, \ldots, y_{n+m}$ of test samples. Here it is assumed that the x-part of test samples are known in advance, which is often referred to as the *transductive setting* [10].

Let us assume that c kernel matrices $K^{[1]}, \ldots, K^{[c]}$ of size $\ell \times \ell$ are available ($\ell := n + m$). Each kernel matrix can be decomposed as

$$K^{[i]} = \begin{pmatrix} L^{[i]} & U^{[i]} \\ (U^{[i]})^\top & Z^{[i]} \end{pmatrix},$$

where $L^{[i]}$ is the kernel matrix between training samples and $U^{[i]}$ is the one between training and test samples. $Z^{[i]}$ is not used in supervised learning tasks.

2.1. TWO WAYS TO MIX

In mixing kernel matrices into one, we consider two ways: *linear mixture* and *nonlinear mixture*. In linear mixture, the kernel matrices are combined as follows:

$$K = \sum_{i=1}^{c} a_i^2 K^{[i]}. \tag{2.1}$$

Here the mixing weights are squared (e.g. a_i^2) to assure positive semidefiniteness of K. The equation (2.1) is decomposed into blocks: $L = \sum_{i=1}^{c} a_i^2 L^{[i]}$ and $U = \sum_{i=1}^{c} a_i^2 U^{[i]}$. As will be shown in Section 2.4, the mixing weights a_i are determined from L only, so we do not need to know $U^{[i]}$'s in advance. Thus this method can as well be applied in non-transductive cases, where test samples are completely unknown.

Let us define a matrix-valued invertible function $f : \Re^{\ell \times \ell} \to \Re^{\ell \times \ell}$. Then (2.1) is easily extended to the *nonlinear mixture*:

$$K = f^{-1}\left(\sum_{i=1}^{c} a_i f(K^{[i]}) \right).$$

There are various choices in f, but we especially focus on $f(M) = M^{1/2}$:

$$K = \left[\sum_{i=1}^{c} a_i (K^{[i]})^{1/2} \right]^2, \tag{2.2}$$

because of computational reasons, i.e. the calculation of $f^{-1}(M) = M^2$ is easily done with matrix-matrix product. Since K is iteratively computed with changing mixing weights, it is desirable to keep the computation of f^{-1} as simple as possible. Another advantage is that the mixing weights may be *negative*, because a squared matrix is always positive semidefinite [10]. Thus, in contrast to the linear mixture, the nonlinear mixture realizes more flexible way of mixing. Notice that we do not deny other possibilities of f – they should be investigated in future works.

2.2. KERNEL FISHER DISCRIMINANT ANALYSIS

After the matrices are mixed, the kernel Fisher discriminant analysis (KFD) [6, 9] is used for estimating labels of test samples. In training, the weight vector is obtained as

$$\boldsymbol{w} = (L + \lambda I)^{-1} \boldsymbol{y}_L,$$

where λ is the regularization parameter, and $\boldsymbol{y}_L$ is the vector of given class labels. The advantage of KFD in comparison to SVM is that the weight vector is described in a closed form. The labels for test data are estimated as

$$\hat{\boldsymbol{y}}_U = \Psi(\boldsymbol{w}^\top U),$$

where $\Psi(\boldsymbol{t}) : \Re^q \to \Re^q$ is a vector valued sign function: when an element of $\boldsymbol{t}$ is positive, it returns 1 in the corresponding element and otherwise -1. The problem is how to determine $\boldsymbol{a} = (a_1, \ldots, a_c)^\top$ and λ such that the number of mismatches between estimated labels and the underlying true labels (i.e., generalization error) is minimized. Since the generalization error is unknown, we will use the *cross validation error* instead. Notice that the parameters $\boldsymbol{a}, \lambda$ are redundant, because their scale does not change the weight vector. We will fix λ to 1 in the following.

2.3. CROSS VALIDATION ERROR

The cross validation error is computed as follows: In r-fold cross validation, the training set is divided into r subsets. The classifier is trained with $r-1$ subsets and then tested with the remaining subset to obtain the classification error. This process is repeated r times and finally the errors are averaged.

Let us divide the training set randomly in r sets of the equal size, and denote by I_j the set of indices in jth subset and by $\bar{I}_j$ the complementary set of I_j. Let us define $\bar{L}_j$ and $\bar{\boldsymbol{y}}_j$ as the kernel matrix and class labels corresponding to $\bar{I}_j$, respectively. The weight vector for I_j is described as

$$\boldsymbol{w}_j = (\bar{L}_j + I)^{-1}\bar{\boldsymbol{y}}_j.$$

The test mismatches for I_j is denoted as

$$\sum_{i \in I_j} \Phi(-y_i \boldsymbol{w}_j^\top \boldsymbol{l}_i^{(j)})$$

where $\boldsymbol{l}_i^{(j)}$ is the kernel vector between ith object and all the objects in $\bar{I}_j$ and $\Phi(t) : \Re \to \Re$ is a step function: when t is positive, it returns 1 and otherwise 0. Thus the cross validation error is described as

$$E(\boldsymbol{a}) = \frac{1}{n}\sum_{j=1}^{r}\sum_{i \in I_j} \Phi(-y_i \boldsymbol{w}_j^\top \boldsymbol{l}_i^{(j)}).$$

2.4. OPTIMIZATION

Our task is to optimize $E(\boldsymbol{a})$ in a gradient-descent manner. Since $E(\boldsymbol{a})$ is not differentiable, we first approximate the step function by a sigmoid function:

$$\phi(t) = \frac{\tanh(\gamma t) + 1}{2},$$

where we set $\gamma = 10$ here. In the following, we are going to calculate the derivatives of the cross validation error. When the derivatives are obtained, $E(\boldsymbol{a})$ can be

optimized by any nonlinear optimization algorithm: Especially we recommend to use some readily-available optimization software to avoid numerical problems. In the experiments of later sections, we used MATLAB's function fminunc, which is an implementation of the sequential quadratic programming method.

The derivative of cross validation error is rewritten as

$$\frac{\partial E}{\partial a_k} = \frac{1}{n}\sum_{j=1}^{r}\sum_{i\in I_j}\frac{\partial E_{ij}}{\partial a_k}, \quad k = 1, \ldots, c, \tag{2.3}$$

where

$$\begin{aligned} E_{ij} &= \phi(-y_i \boldsymbol{w}_j^\top \mathbf{l}_i^{(j)}) \\ &= \phi(-y_i \bar{\boldsymbol{y}}_j^\top (\bar{L}_j + I)^{-1} \boldsymbol{l}_i^{(j)}). \end{aligned}$$

Due to the chain rule, we have

$$\frac{\partial E_{ij}}{\partial a_k} = \sum_{s=1}^{n}\sum_{t=s}^{n}\frac{\partial E_{ij}}{\partial L_{st}}\frac{\partial L_{st}}{\partial a_k}.$$

Thus we need to calculate $\frac{\partial E_{ij}}{\partial L_{st}}$ and $\frac{\partial L_{st}}{\partial a_k}$. Let us start with the latter because it is simpler: For the linear mixture case (2.1), it turns out that

$$\frac{\partial L_{st}}{\partial a_k} = 2a_k L_{st}^{[k]}.$$

For the nonlinear mixture case (2.2), we have

$$\frac{\partial L_{st}}{\partial a_k} = 2\sum_{l=1}^{c} a_l (\boldsymbol{s}_s^{[k]})^\top \boldsymbol{s}_t^{[l]},$$

where $\boldsymbol{s}_i^{[k]}$ is the ith column of $(K^{[k]})^{1/2}$. On the other hand, the calculation of $\frac{\partial E_{ij}}{\partial L_{st}}$ has three options.

- When L_{st} corresponds to the uth element of $\boldsymbol{l}_i^{(j)}$, $\frac{\partial E_{ij}}{\partial L_{st}}$ is obtained as the uth element of the following vector:

  $$\frac{\partial E_{ij}}{\partial l_i^{(j)}} = \phi'(-y_i \bar{\boldsymbol{y}}_j^\top (\bar{L}_j + I)^{-1} \boldsymbol{l}_i^{(j)})[-y_i (\bar{L}_j + I)^{-1} \bar{\boldsymbol{y}}_j]$$

  where $\phi'(t) = \frac{\gamma}{2}\operatorname{sech}^2(\gamma t)$.
- When L_{st} corresponds to the (u, v) element of $\bar{L}_j$, $\frac{\partial E_{ij}}{\partial L_{st}}$ is obtained as the (u, v) element of the following matrix:

  $$\frac{\partial E_{ij}}{\partial L_j} = \phi'(-y_i \bar{\boldsymbol{y}}_j^\top (\bar{L}_j + I)^{-1} \boldsymbol{l}_i^{(j)})[y_i (\bar{L}_j + I)^{-1} \bar{\boldsymbol{y}}_j][(\bar{L}_j + I)^{-1} \boldsymbol{l}_i^{(j)}]^\top.$$

- Otherwise, $\frac{\partial E_{ij}}{\partial L_{st}} = 0$.

Finally the derivative (2.3) can be computed by assembling all the results.

3. Experiments

3.1. BACTERIA CLASSIFICATION

In this section, we perform a supervised classification experiment for classifying bacteria based on two marker amino acid sequences: gyrase subuit B (gyrB) protein and RNA polymerase sigma factor type D (rpoD). Gyrase is a type II DNA topoisomerase which is an enzyme that controls and modifies the topological states of DNA supercoils. This protein is known to be well preserved over evolutional history among bacterial organisms thus is supposed to be a better identifier for evolutionary distance measurement than the popular 16S rRNA [5]. RNA polymerase sigma factor is a subunit of RNA polymerase which plays essential role for RNA synthesis in cells. The sigma factor recognizes promoter signal in a DNA sequence and points where to start transcription. The sigma factors are classified into several categories due to promoter sequences they recognize. rpoD (σ^{70}) is a major sigma factor because promoters recognized by rpoD regulates many of genes in *E. coli* and other bacteria. Although gyrB and rpoD play quite different roles, they both are crucial to their organisms.

The dataset used in the experiment has 48 pairs of gyrB and rpoD amino acid sequences of three genera (*Pseudomonas luorescens*:15, *Pseudomonas marginalis*:13, *Pseudomonas putida*:20). For simplicity, let us call these genera as class 1–3, respectively. For gyrB and rpoD, we computed the second order count kernel, which is the dot product of bimer counts [12]. We performed the classification experiment with four different approaches: gyrB alone, rpoD alone, linear mixture (i.e., Lmix), and nonlinear mixture (i.e., NLmix). In order to compare these approaches, three two-class problems (i.e., 1–2, 1–3, 2–3) are devised by coupling two classes. For supervised learning, the dataset is randomly divided to 80% training and 20% test sets, and the experiment is iterated 20 times with different training/test splits. Notice that we used the 5-fold cross validation error in all approaches. In single kernel cases, only the regularization parameter is optimized to minimize the cross validation error.

The mean error rates are listed in Table 1. In the first two cases (i.e., 1–2 and 1–3), the nonlinear mixture significantly outperformed single kernels and the linear

Table 1. Mean test errors of bacteria classification with standard deviation in (·). The classes 1–3 correspond to *Pseudomonas luorescens*, *Pseudomonas marginalis* and *Pseudomonas putida*, respectively. The labels 'Lmix' and 'NLmix' denote the linear mixture of kernels and the nonlinear mixture of kernels, respectively. The best results are highlighted with bold face.

	gyrB	rpoD	Lmix	NLmix
1–2	35.00 (21.40)	19.00 (18.89)	19.00 (21.00)	17.00 (21.79)
1–3	14.29 (13.11)	3.57 (7.86)	3.57 (6.35)	2.86 (5.86)
2–3	5.00 (7.84)	1.67 (7.45)	0.00 (0.00)	0.83 (3.73)

Table 2. Cross validation errors of bacteria classification with standard deviation in (·).

	gyrB	rpoD	Lmix	NLmix
1–2	31.85 (9.81)	16.36 (5.58)	16.32 (5.72)	9.68 (6.72)
1–3	8.67 (3.12)	4.66 (3.31)	5.23 (3.35)	3.73 (2.33)
2–3	6.69 (2.12)	1.03 (1.56)	0.43 (0.43)	0.34 (0.40)

mixture. In the last case (i.e., 2–3), the linear mixture was the best. In order to assure that the optimization process did not stuck at notorious local minima, we will also show the cross validation errors in Table 2. As seen in the table, the nonlinear mixture always achieved smaller cross validation error than the linear mixture. Probably the reason is that the nonlinear mixture provides more diverse choices because it allows negative mixing weights.

3.2. GENE FUNCTION PREDICTION

The second experiment focuses on predicting the function of yeast genomes. We utilize two kinds of data: phylogenetic profiles and amino acid sequences. The phylogenetic profile is a d-dimensional vector which represents phylogenetic relations between a gene of a certain organism and genomes of d different organisms. The ith element is one if the ith genome has a homologous gene. Otherwise it is zero. We used the same phylogenetic profiles used by Pavlidis *et al.* [7] (available at http://www.cs.columbia.edu/compbio/exp-phylo/) where, instead of one and zero, negative logarithm of the lowest E-value reported by BLAST version 2.0 is used (negative values correspond to E-value > 0). For each yeast gene, 24 bacterial genomes are used for constructing phylogenetic profiles. The gene expression data is also available from this site, but we did not use it because it did not significantly help improving classification performance (see [7]). The sequence data is obtained from Comprehensive Yeast Genome Database (CYGD) (http://mips.gsf.de/proj/yeast/).

Classification experiments are carried out using gene functional categories from CYGD. We especially focused on the category 'Transport Facilitation' and performed classification with respect to its subcategories: *ion transporters*, *C-compound and carbohydrate transporters*, *amino-acid transporters*, *transport ATPases*, *ABC transporters*. We denote them as class 1 to 5, respectively. There are other subcategories, but we excluded them because the number of genes is too small. Notice that we could not use the genes whose sequence or phylogenetic profile is not included in the databases. As a result, the number of genes in each class is listed as 60, 32, 22, 36 and 16. As in the previous section, we devised 10 problems by coupling two classes. The dataset is randomly divided to 80% training and 20% test sets, and the experiment is iterated 20 times with different training/test splits. We again used the 5-fold cross validation error here.

Table 3. Test errors in gene function prediction. The classes 1–5 corresponds to *ion transporters*, *C-compound and carbohydrate transporters*, *amino acid transporters*, *transport ATPases*, *ABC transporters*. 'Phy' and 'Seq' denote the kernel matrices of phylogenetic profiles and gene sequences, respectively. 'Phy^2' and 'Seq^2' are the second order polynomial kernels of 'Phy' and 'Seq', respectively.

	Phy	Seq	Phy^2	Seq^2	Lmix	NLmix
1–2	17.22 (7.41)	19.17 (11.88)	15.00 (8.28)	17.78 (9.64)	12.50 (8.62)	14.44 (7.30)
1–3	13.75 (6.91)	9.06 (7.71)	3.75 (3.74)	9.38 (7.72)	4.38 (4.11)	4.38 (4.11)
1–4	42.11 (19.01)	39.21 (12.13)	43.68 (11.21)	45.00 (10.31)	43.95 (12.23)	42.63 (12.54)
1–5	28.33 (9.64)	23.33 (9.05)	7.67 (6.22)	23.67 (8.78)	7.67 (5.83)	8.33 (5.67)
2–3	5.00 (8.89)	7.50 (8.51)	6.00 (7.54)	7.50 (8.51)	3.00 (5.71)	4.50 (6.05)
2–4	19.23 (11.02)	17.69 (8.68)	8.85 (7.60)	15.00 (7.68)	6.92 (7.01)	8.46 (8.24)
2–5	10.56 (7.63)	10.56 (11.10)	5.00 (7.63)	7.78 (8.90)	5.00 (6.72)	3.33 (6.35)
3–4	14.09 (12.33)	5.00 (5.50)	2.73 (5.19)	3.64 (5.44)	2.73 (5.19)	3.64 (6.19)
3–5	3.57 (6.35)	0.00 (0.00)	3.57 (6.35)	0.00 (0.00)	3.57 (6.35)	3.57 (6.35)
4–5	11.00 (9.12)	23.50 (18.14)	3.50 (4.89)	12.50 (12.09)	3.00 (4.70)	4.00 (5.98)

The normalized dot product is adopted for the kernel function of phylogenetic profiles:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \frac{\boldsymbol{x}^\top \boldsymbol{x}'}{\|\boldsymbol{x}\| \|\boldsymbol{x}'\|}$$

where $\boldsymbol{x}$ and $\boldsymbol{x}'$ denote the vector of phylogenetic profile. For gene sequences, we used the second order count kernel as in the previous experiment. In order to show that our methods work with more than two matrices, we additionally constructed the second order polynomial kernels [10] for both phylogenetic profiles and sequences:

$$K'_{ij} = (K_{ij} + 1)^2.$$

Finally we have four kernel matrices in total.

Table 4. Cross validation errors in gene function prediction.

	Phy	Seq	Phy^2	Seq^2	Lmix	NLmix
1–2	24.77 (8.35)	18.92 (8.60)	15.95 (6.35)	18.47 (8.05)	11.85 (6.39)	9.68 (6.49)
1–3	17.23 (7.13)	25.26 (5.62)	9.16 (4.92)	21.78 (5.81)	7.75 (4.43)	6.61 (4.46)
1–4	40.07 (10.21)	32.39 (8.41)	40.29 (10.27)	33.85 (9.06)	30.51 (7.78)	26.64 (8.51)
1–5	22.88 (8.93)	19.65 (7.19)	13.61 (4.76)	18.28 (6.87)	12.60 (4.78)	10.61 (5.12)
2–3	6.66 (4.81)	16.04 (11.24)	7.42 (4.93)	16.36 (11.31)	6.08 (4.53)	3.97 (4.22)
2–4	20.22 (9.58)	16.40 (6.88)	15.68 (8.98)	16.73 (6.80)	9.41 (5.51)	8.01 (4.72)
2–5	7.00 (7.28)	15.96 (8.57)	5.66 (5.24)	14.67 (8.89)	5.90 (5.76)	4.66 (4.06)
3–4	10.04 (6.05)	9.86 (7.92)	7.48 (5.36)	10.62 (7.92)	4.64 (5.96)	2.64 (4.35)
3–5	8.50 (7.51)	10.97 (9.64)	8.29 (6.97)	10.92 (9.53)	5.39 (5.20)	4.94 (5.15)
4–5	24.57 (11.01)	30.66 (10.85)	14.40 (9.45)	27.86 (10.03)	13.50 (8.00)	8.28 (7.46)

The test errors are listed in Table 3. The mixing approaches outperformed single kernels in 6 out of 10 experiments. In 1–3, 1–5 and 3–5, one of the kernels performed exceptionally well, so mixing with other 'poor' kernels did not lead to improve performance. In 1–4, the error rates of all kernels are very poor (i.e., close to 50%), so the comparison between error rates is not really meaningful.

When looking at the cross validation errors (Table 4), the nonlinear mixture achieves the best result in all experiments. Thus, in most experiments, the best method in cross validation error was not the best in test error. This phenomenon may be termed as 'overfitting to cross validation error'. When the number of samples is small, the cross validation error is likely to have such a large variance that its behavior is significantly different from test error. As a result, minimizing the cross validation error becomes misleading. This problem is not unique to cross validation and virtually every model selection criterion suffers from large variance in small sample cases [1]. If some countermeasure is taken against overfitting (e.g., feature selection or regularization on cross validation error), the test error of nonlinear mixture would get better. However it is not tried yet at this point.

4. Conclusion

In this paper, we proposed two ways to mix kernel matrices to minimize the cross validation error and applied them to biological experiments with promising results. Here we only dealt with two class problems to keep the computation of cross validation error reasonably simple. The extension to multiclass problems is in principle possible but the computation of cross validation error and its derivatives would be more complicated.

In gene function prediction, we encountered the problem of overfitting to the cross validation error, because of small sample sets and high dimensionality. Such difficult situations have seldom been considered in the conventional statistical community, but bioinformatic applications are recently urging to develop new robust approaches (e.g., [11]). The kernel mixing is regarded as a new challenging problem, for which robust methods are definitely required.

Our methods can be applied to any type of kernels derived from any data. In future works, we are interested in applying our methods to the diffusion kernels derived from metabolic network [13]. Gene expression [4] is an especially interesting target as well.

Acknowledgements

The authors gratefully acknowledge that the bacterial amino acid sequences are offered by courtesy of Identification and Classification of Bacteria (ICB) database team [14]. The authors would like to thank J.-P. Vert and T. Tsuchiya for fruitful discussions.

References

1. Burnham, K. P. and Anderson, D. R.: *Model Selection and Inference: A Practical Information–Theoretic Approach* Springer, 1998.
2. Chapelle, O., Vapnik, V., Bousquet, O. and Mukherjee, S.: Choosing multiple parameters for support vector machines, *Machine Learning* **46**(1–3) (2002), 131–159.
3. Duan, K., Keethi, S. S. and Poo, A. N.: Evaluation of simple performance measures for tuning svm hyperparameters. Technical Report CD-01-11, Control Division, Dept. of Mechanical Engineering, National University of Singapore, 2001.
4. Hanisch, D., Zien, A., Zimmer, R. and Lengauer, T.: Co-clustering of biological networks and gene expression data, *Bioinformatics* **18** (2002), S145–S154.
5. Kasai, H., Bairoch, A., Watanabe, K., Isono, K., Harayama, S., Gasteiger, E. and Yamamoto, S.: Construction of the *gyrb* database for the identification and classification of bacteria, In: *Genome Informatics, 1998*, pp. 13–21, Universal Academic Press, 1998.
6. Mika, S., Rätsch, G., Weston, J., Schölkopf, B. and Müller, K.-R.: Fisher discriminant analysis with kernels, In: Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas (eds), *Neural Networks for Signal Processing IX*, pp. 41–48. IEEE, 1999.
7. Pavlidis, P., Weston, J., Cai, J. and Grundy, W. N.: Gene functional classification from heterogeneous data, In: *Proc. RECOMB2001*, pp. 242–248, 2001.
8. Pellegrini, M., Marcotte, E. M., Thompson, M. J., Eisenberg, D. and Yeates, T. O.: Assigning protein functions by comparative genome analysis: Protein phylogenetic profiles, *Proc. Natl. Acad. Sci. USA* **96**(8) (1999), 4285–4288.
9. Roth, V. and Steinhage, V.: Nonlinear discriminant analysis using kernel functions. In: S. A. Solla, T. K. Leen and K.-R. Müller (eds), *Advances in Neural Information Processing Systems 12*, pp. 568–574, MIT Press, 2000.
10. Schölkopf, B. and Smola, A. J.: *Learning with Kernels* MIT Press, Cambridge, MA, 2001.
11. Schölkopf, B., Weston, J., Eskin, E., Leslie, C. and Noble, W. S.: A kernel approach for learning from almost orthogonal patterns, In: *Proc. 13th European Conference on Machine Learning*, pp. 511–528, 2002.
12. Tsuda, K., Kin, T. and Asai, K.: Marginalized kernels for biological sequences, *Bioinformatics* **18** (2002), S268–S275.
13. Vert, J.-P. and Kanehisa, M.: Graph-driven features extraction from microarray data. Technical Report physics/0206055, Arxiv, June 2002.
14. Watanabe, K., Nelson, J. S., Harayama, S. and Kasai, H.: ICB database: the gyrB database for identification and classification of bacteria, *Nucleic Acids Res.* **29** (2001), 344–345.
15. Yamamoto, S., Kasai, H., Arnold, D. L., Jackson, R. W., Vivian, A. and Harayama, S.: Phylogeny of the genus Pseudomonas: intrageneric structure reconstructed from the nucleotide sequences of gyrB and rpoD genes, *Microbiology* **146** (2000), 2385–2394.