Modifying Kernels Using Label Information Improves SVM Classification Performance

Renqiang Min and Anthony Bonner
Department of Computer Science
University of Toronto
Toronto, ON M5S3G4, Canada
minrq@cs.toronto.edu

Zhaolei Zhang
Banting and Best Department of
Medical Research, University of Toronto
Toronto, ON M5G1L6
Zhaolei.Zhang@utoronto.ca

Abstract

Kernel learning methods based on kernel alignment with semidefinite programming (SDP) are often memory intensive and computationally expensive, thus often impractical for problems with large-size dataset. We propose a method using label information to modify kernels based on SVD and a linear mapping. As a result, the new kernel matrix reflects the label-dependent separability of the data in a better way than the original kernel matrix. In addition, our experimental results on USPS handwritten digits and the SCOP dataset, show that the SVM classifier based on the improved kernels has better performance than the SVM classifier based on the original kernels; moreover, SVM based on the improved Profile kernel with pull-in homologs (see experiment section for explanations) produced the best results for remote homology detection on the SCOP dataset compared to the published results.

1. Introduction

Protein sequence classification and fold recognition is still a challenging task in the bioinformatics research community. Generative models (e.g., profile HMMs [3] and [4]) and discriminative models (e.g., kernel SVMs [1] and [6]) have been applied to solve this problem.

It has been shown that the kernel SVM method has better classification and prediction performance on protein sequence data than some other methods (see [1] and [6]). Several kernels such as pairwise-sequence-similarity-score based kernels and mismatch-string kernels, which are especially suitable for protein sequence data that consists of a limited number of letters from the amino acid alphabet, are frequently used in sequence classification and structure prediction. However, the label information of labeled sequences (the class membership of the data points; in a binary classification problem, the label of a data point is 1 or 0) is completely or partly ignored in the construction process of these kernels, and the available information between pairwise unlabeled sequences is also ignored both in the training phase and in the testing phase. In this paper, we will incorporate the label information of training data into the construction process of a new kernel, hoping that the obtained kernel reflects the real neighborhood property of the data in a better way than the original kernel. We believe this helps classification in most situations.

In this paper, we solve protein remote homology detection problem and handwritten digit classification problem to evaluate the performance of our proposed approach. We will use RBF kernels as base kernels for handwritten digit classification and two mismatch-string kernels as base kernels [6] for protein classification. We present our experimental results for digit classification and protein homology detection on the SCOP dataset in Section 5. In Section 6, we conclude the paper with some discussions.

2 SVM based on mismatch-string kernel

In kernel SVM, by constructing a kernel function, we can map every data point, x_i, to a high-dimensional feature space, in which an SVM can be used to generate a separating hyperplane. Moreover, we can solve the dual problem of the original optimization problem and then perform all the calculations using the kernel trick.

Kernel function can map sequences consisting of letters to a high-dimensional numerical space. For example, suppose that A is an alphabet with ℓ symbols ($\ell = 20$ for protein sequences). A k-mer string kernel maps every sequence in A to a ℓ^k-dimensional feature space in which coordinates are indexed by all possible sub-sequences of length k (k-mers). The mismatch string kernel extends this idea with a tolerance of at most m mismatches when counting the number of occurrences of a k-mer in an input sequence. A

Profile Kernel [2] extends the mismatch-string kernel by using additional profile information of each sequence, that is, the emission probability of every amino acid at each position in respective sequences. Instead of treating all k-mers with less than m mismatches the same like the mismatch-string kernel, the profile-kernel examines these k-mers further by looking at the emission probabilities at the mismatch positions and only tolerates some types of mismatches by thresholding. Note that all entries in these kernel matrices are non-negative. In fact, for a wide range of applications including protein classification and handwritten digit classification, the components of the constructed kernels are positive. In our method, we require the kernel entries be positive.

3 Related methods

A kernel matrix K with $K(i,j) = \Phi(i)^T\Phi(j)$ can be used to derive a similarity matrix based on square Euclidean distances between any pairwise data points, i and j, in the feature space, as follows:

$$Dist^2(i,j) = K(i,i) + K(j,j) - 2K(i,j) \qquad (1)$$

Although SVM generates the optimal separating hyperplane in the feature space given a specific kernel, it does not adjust the given kernel and make it more discriminative. Therefore, it leaves room for improvement as we can apply the aforementioned idea of preserving neighbor identity to construct new kernels in order to achieve better separability in the new feature space. Instead of computing more discriminative features of data points explicitly, we can construct a more discriminative kernel directly and all the computations needed by training and classification can be cast onto the new kernel matrix. As discussed in [7] and [8], a linear combination of some predefined kernels is used to generate new kernels, and the mixing coefficients are calculated by aligning the training part of the combined kernel to the training part of an optimal kernel K as follows:

$$K = \begin{bmatrix} K_{tr}^{opt} & {K_{tt}}^T \\ K_{tt} & \text{unused} \end{bmatrix} \qquad (2)$$

$$K_{tr}^{opt}(i,j) = \begin{cases} +1 & \text{if label}(i) = \text{label}(j) \\ -1 & \text{otherwise} \end{cases} \qquad (3)$$

where i and j index data points in the training set, and tr and tt respectively denote the training part and the test part (this rule applies to all the denotations in the paper). If there are n training data points and m test data points, K_{tr} is an n-by-n block sub-matrix and K_{tt} is an m-by-n block sub-matrix in K. In fact, doing kernel alignment is to make the constructed kernel approximate the neighbor identity and data separability reflected by the optimal kernel. From Equation (1), we can easily find that the optimal kernel makes the distances between pairwise data points having the same label be 0 and the distances between pairwise data points having the different labels be 2. That is to say, the kernel alignment algorithms actually use the label information to construct a new kernel to achieve good data potability. However, doing the alignment to calculate the mixing coefficients costs a lot of memory and is very computationally expensive or impossible for handling large datasets for combining many kernels. In [9], label information is used to learn a linear transformation matrix in a high-dimensional feature space to generate a new distance metric using kernels. However, linear transformation in original feature space cannot improve linear classifiers like SVM. Therefore, their method essentially differs from our method for improving SVM.

In this paper, we propose another efficient approach for constructing new kernels using label information, which is based on scaling, matrix decomposition, and a linear mapping, to achieve better data separability as discussed above. The approach is easy to implement and easy to extend to many types of kernels.

4 Improved kernels using label information

Suppose that we have a dataset as described in Section 2 (we only consider the two-class problem here) and a given mapping from the input data space to a high dimensional feature space. We can then construct a kernel K based on the mapping.

Given the constructed kernel K with $K(i,j) = \Phi(i)^T\Phi(j)$ and the label information of training data, we want a new kernel that better reflects the neighbor identity and separability of the data consistent to the current labels of training data. If two arbitrary data points in the training set, i and j, have the same label, we multiply the inner product of their feature vectors by a scaling factor, γ (see Section 5 for detailed discussion about choosing γ), which is greater than 1, to get a new kernel matrix $\hat{K}$ as follows:

$$\hat{K} = \begin{bmatrix} \hat{K}_{tr} & \hat{K}_{tt}^T \\ \hat{K}_{tt} & \text{unused} \end{bmatrix} \qquad (4)$$

where

$$\hat{K}_{tr}(i,j) = \begin{cases} \gamma K_{tr}(i,j) & \text{if label}(i) = \text{label}(j) \\ K_{tr}(i,j) & \text{otherwise} \end{cases} \qquad (5)$$

The resulting kernel matrix $\hat{K}$ have larger kernel alignment score (KAS) than K [7]. $KAS(\hat{K}_{tr}) = \frac{\sum_{ij} \hat{K}_{ij} y_i y_j}{n\sqrt{\sum_{ij} \hat{K}_{ij}^2}} > KAS(K_{tr})$, where n is the size of the training set, $y_i, y_j \in \{+1, -1\}$, and $i, j = 1, \ldots, n$. This inequality can be easily proved by dividing the numerator and

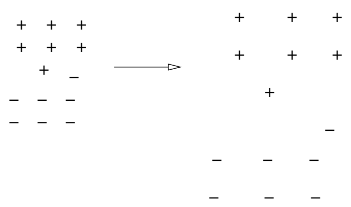

Figure 1. The data distribution in the original feature space and in the new feature space. '+' means positive and '-' means negative.

the denominator of $KAS(\hat{K}_{tr})$ by γ. Note that we only use the label information of the training set to modify the training part of the kernel matrix here. This modification will affect both the training part and the test part of K. The test part $\hat{K}_{tt}$ of the new kernel matrix $\hat{K}$ is calculated using kernel extrapolation, which is based on a linear mapping. The matrix $\hat{K}_{tr}$ is positive semidefinite. The distances between pairwise data points in the new feature space corresponding to $\hat{K}_{tr}$ are as follows:

$$\hat{Dist}^2_{tr}(i,j) = \begin{cases} \gamma Dist^2(i,j) & \text{if label}(i) = \text{label}(j) \\ \gamma Dist^2(i,j) + & \text{otherwise.} \\ 2(\gamma-1)K_{tr}(i,j) & \gamma > 1 \end{cases} \tag{6}$$

Here $Dist^2(i,j)$ is defined in Equation (1), and i and j index the data points in the training set. We see from Equation (6) that, in the new feature space, the distance between points having the same label is increased by a factor of γ. Moreover, since $\gamma > 1$ and $K_{tr}(i,j)$ is non-negative, the distance between points having different labels is increased even further by the additional term $2(\gamma-1)K_{tr}(i,j)$. That is to say, in the feature space defined by $\hat{K}_{tr}$, data points with the same label stay relatively close together, while data points with different labels move relatively further apart. Figure 1 illustrates the separation of data points in feature spaces corresponding to K_{tr} and $\hat{K}_{tr}$.

We can also interpret the similarity between a pair of data points, i and j, in terms of the angle between their feature vectors, $\Phi(i)$ and $\Phi(j)$, which is given by $\theta = \arccos[K(i,j)/\sqrt{K(i,i)K(j,j)}]$. The angle between two points with the same label is the same in the new feature space and the original feature space, while the angle between two points with different labels is larger in the new feature space than in the original feature space. This can also be seen in Figure 1.

After the training part, $\hat{K}_{tr}$, of the new kernel is constructed, we need to estimate the testing part, $\hat{K}_{tt}$. That is, to classify a test case by an SVM based on $\hat{K}$, we need to estimate the inner products of the new feature vector of the test case and the new feature vectors of all the training cases. This can be done by approximating all the high-dimensional feature vectors by lower, N-dimensional feature vectors, where N is the size of the training set. To do this, we decompose the training part of K and $\hat{K}$, denoted by K_{tr} and $\hat{K}_{tr}$, respectively, into SVD form as follows:

$$K_{tr}V_{tr} = V_{tr}D_{tr} \tag{7}$$

$$K_{tr} = V_{tr}D_{tr}{V_{tr}}^T = {W_{tr}}^T W_{tr} \tag{8}$$

where

$$W_{tr} = (V_{tr}\sqrt{D_{tr}})^T \tag{9}$$

Similarly,

$$\hat{K}_{tr}\hat{V}_{tr} = \hat{V}_{tr}\hat{D}_{tr} \tag{10}$$

$$\hat{K}_{tr} = \hat{V}_{tr}\hat{D}_{tr}\hat{V}_{tr}^T = \hat{W}_{tr}^T\hat{W}_{tr} \tag{11}$$

where

$$\hat{W}_{tr} = (\hat{V}_{tr}\sqrt{\hat{D}_{tr}})^T \tag{12}$$

In these equations, the columns of V_{tr} are orthogonal eigenvectors of K_{tr}^2, and D_{tr} is a diagonal matrix containing the corresponding eigenvalues. Likewise for $\hat{V}_{tr}$, $\hat{K}_{tr}$ and $\hat{D}_{tr}$. W_{tr} and $\hat{W}_{tr}$ are n-by-n matrices, where n is the size of the training set.

We can view W_{tr} as a compressed representation of the high-dimensional feature vectors of the training data in a lower dimensional space. Note that this representation preserves all the inner products. We can interpret $\hat{W}_{tr}$ in the same way. Moreover, the new kernel, $\hat{K}_{tr}$, can be calculated from $\hat{W}_{tr}$, which in turn can be computed by applying a linear transformation to W_{tr}, as the following lemma shows:

Lemma 1 *$AW_{tr} = \hat{W}_{tr}$, where $A = \sqrt{\hat{D}_{tr}}\hat{V}_{tr}^T V_{tr}\frac{1}{\sqrt{D_{tr}}}$*

Here, the expression $\frac{1}{\sqrt{D}}$ means the inverse of the diagonal matrix $\sqrt{D}$. The lemma itself follows immediately from equations (9) and (12). We interpret this lemma as follows: K_{tr} and $\hat{K}_{tr}$, respectively, corresponds to feature space F and $\hat{F}$ with W_{tr} lying in F and $\hat{W}_{tr}$ lying in $\hat{F}$; there exists a linear transformation between F and $\hat{F}$. We shall use the linear transformation, A, to estimate the matrix $\hat{K}_{tt}$, the testing part of $\hat{K}$. This involves the following assumption:

Assumption 1 *The linear relation shown in Lemma 1 can be extended to $A[W_{tr}; W_{tt}] = [\hat{W}_{tr}; \hat{W}_{tt}]$, where W_{tt} and $\hat{W}_{tt}$ are m-by-n matrices which satisfy $W_{tt}^T W_{tr} = K_{tt}$ and $\hat{W}_{tt}^T\hat{W}_{tr} = \hat{K}_{tt}$, n and m are respectively the size of the training set and the test set.*

In this assumption, we assume that: W_{tt} lies in F and $\hat{W}_{tt}$ lies in $\hat{F}$; applying the linear transformation A to W_{tt} will result in the n-dimensional feature vectors of test data $\hat{W}_{tt}$ in the reduced feature space $\hat{F}$, which better reflects the label-dependent separability of the test data points as A does to W_{tr}. The value of this assumption is tested empirically in Section 5, where we show that the resulting kernel leads to an SVM classifier with significantly improved performance.

Note that W_{tt} and $\hat{W}_{tt}$ are N-dimensional feature vectors representing the test data points[1]. Using Lemma 1 and Assumption 1, we can calculate $\hat{K}_{tt}$. First, from the definitions of K and W,

$$W_{tt}^T W_{tr} = K_{tt} \tag{13}$$

and so by equation (9),

$$W_{tt} = \frac{1}{\sqrt{D_{tr}}} V_{tr}^T K_{tt}^T \tag{14}$$

According to Assumption 1, we have

$$\hat{K}_{tt} = \hat{W}_{tt}^T \hat{W}_{tr} = K_{tt}(V_{tr} \frac{1}{D_{tr}} V_{tr}^T)\hat{K}_{tr} = K_{tt} K_{tr}^{-1} \hat{K}_{tr} \tag{15}$$

When calculating $\hat{K}_{tt}$, we need to calculate K_{tr}^{-1} first, and then we can obtain $\hat{K}_{tt}$ easily by Equation (15). Note that we need not perform SVD at all and the inverse of K_{tr} can be computed in Matlab very fast (it takes less than 10 seconds to get the inverse of a 2620-by-2620 kernel matrix in our machine with 3.0GHz CPU and 4.0GB memory)[2]. After the new kernel is constructed, we can apply machine learning techniques based on the kernel to classification, clustering, or regression problems.

5 Experimental results

5.1 Experiments on handwritten digit classification

We use USPS handwritten digits to perform binary classification to test the performance of our proposed approach. In the dataset, there are 1100 images with 256 gray-scale pixels for each of the ten digits. We choose some similar digits to compose pairs for binary classification to increase the difficulty of the tasks. For each binary classification task, we randomly choose 600 images for training and 500 images for testing for each image; then we test the performance of the base kernel and the modified kernel on this dataset; and we repeat this procedure 5 times. In the experiments, RBF kernels are used as base kernels, in which the free parameter σ is chosen using 5-fold Cross Validation (CV). The free parameter C in SVM and the free parameter γ are also chosen using 5-fold CV. We used a greedy approach to choose the free parameters to avoid a grid search. The results are summarized in Table 5.1. We can see that the modified kernels consistently give better performance than the original RBF kernels.

[1]Note that our method is not transductive and has wider applicability in practice than transductive methods.

[2]Equation (15) requires K_{tr} is non-singular, and if it is singular, it means that some rows in K_{tr} corresponding to some training data points can be expressed as the linear combination of some other rows in K_{tr}, we can simply remove the redundant rows to get a non-singular K_{tr} or set K_{tr} to be $K_{tr} + \epsilon I$.

USPS handwritten digit classification results

	run1	run2	run3	run4	run5
8vs9	11	11	11	12	12
8vs9 +	8	8	7	6	7
0vs6	6	7	6	7	6
0vs6 +	4	4	4	4	3
4vs6	13	13	11	12	13
4vs6 +	11	10	7	9	11

Table 1. '8vs9' and '8vs9+' represent the binary classification task for the pair '8' versus '9'. Similarly, the rest rows correspond to the results for other different pairs. The rows with '+' correspond to the modified kernels using label information and the rest rows correspond to the base kernels (RBF). The integer numbers in the right five columns inside the table are the number of mis-classifications. Different runs denote different random splittings of the dataset for training and testing.

5.2 Experiments on protein remote homology detection

We also determine the classification performance of the new kernels against the original kernels by comparing their ability to detect protein remote homology. A benchmark dataset, which was derived by Jaakkola from the SCOP database (see [5] and [1]), is used here. In SCOP, protein sequences are classified into a 4-level hierarchy: class, fold, superfamily, and family, starting from the top. Remote homology is simulated by choosing all the members of a family as positive test data, some family (or families) in the same superfamily of the test data as positive training data, all sequences outside the fold of the test data as either negative training data or negative test data, and sequences that are neither in the training set nor in the test set are considered as unlabelled data. This data splitting scheme has been used in several previous papers (see [1] and [6]). We used the same training and test data split as those used in [6]. The version 1.59 of the SCOP dataset from http://astral.berkeley.edu is used, in which no pair of sequences share more than 95% identity.

In the experiments, there are 54 target test families altogether classified into four classes: alpha proteins, beta proteins, alpha and beta proteins, and small proteins. In the

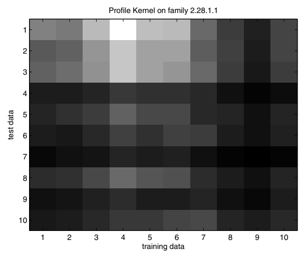

Figure 2. A block sub-matrix in K_{tt} of the original Profile Kernel[PSIBLAST] on family 2.28.1.1 (Legume lectins). See the text for explanation.

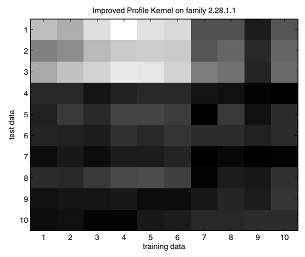

Figure 3. A block sub-matrix in $\hat{K}_{tt}$ of the improved Profile Kernel[PSIBLAST] on family 2.28.1.1 (Legume lectins). See the text for explanation.

data splits, for most experiments, there are only several positive test cases but hundreds or even thousands of negative test cases. The maximum number of positive test cases is below 30, but the maximum number of negative test cases is above 2600. The minimum number of positive test case is 1, but the minimum number of negative test cases is still above 250. So, in the experiments with a very limited number of positive test cases and a large number of negative test cases, we can almost ignore the ranking of positive cases below 50 negative cases. In such situations, we consider that the ROC_{50} score is much more important than the ROC score. Here, a ROC curve plots the rate of true positives as a function of the rate of false positives at different decision thresholds, the ROC score is the area under the curve, and the ROC_{50} score is the ROC score computed up to the first 50 false positives. Thus, in our experiments, we only compare the ROC_{50} scores corresponding to different kernels.

Because our approach to generating new kernels based on label information is independent of given kernels, we choose two representative kernels, which were, respectively, a 'Mismatch kernel + homologs [PSI-BLAST]' as described in [6] and a 'Profile kernel' as described in [2] also 'plus homologs [PSI-BLAST]' as base kernels. 'kernels + homologs [PSI-BLAST]' refers to a semi-supervised learning method: prior to training SVM, close homologs of the training data in the unlabelled set found by PSI-BLAST with E-value less than 0.05 are added to the positive training set, and are labeled as positive (we call this 'pull-in homologs'). We choose the first kernel because it gives the best results on remote homology detection among the kernels that don't use the profile information; and we choose the second kernel because it produced the best results on SCOP among all the kernels (we don't consider transductive learning in this paper). To perform SVM classification based on the kernels, we used the SVM classifier in the freely available Spider Matlab machine learning package.

We compared the methods using the mismatch kernel

Mean ROC_{50} scores over different protein classes

	Alpha Proteins	Beta Proteins	Alpha and Beta Proteins	Small Proteins
K1	0.4874	0.5208	0.5798	0.5905
ImproK1	0.5395	0.5283	0.5933	0.6010
K2	0.7909	0.8156	0.8924	0.8687
ImproK2	0.8172	0.8276	0.9075	0.8808

Table 2. 'K1' represents 'Mismatch kernel + [PSI-BLAST]', 'ImproK1' represents 'Improved Mismatch kernel + [PSI-BLAST]', 'K2' represents 'Profile Kernel + [PSI-BLAST]', and 'ImproK2' represents 'Improved Profile Kernel + [PSI-BLAST]'.

with $k = 5$ and $m = 1$ and the profile kernel with $k = 5$ and $\sigma = 7.5$, and the γ is set by CV. In the experiments in which the number of positive training cases is greater than or equal to 5, we respectively generated a random permutation of the positive training cases and of the negative training cases, then we divided the two permutations into 5 folds denoted by P_i and N_i, $i = 1, \cdots, 5$. We form a new set $M = \{\{P_i, N_i\} | i = 1, \cdots, 5\}$, then we did 5-fold CV on M and chose γ corresponding to the biggest mean ROC_{50} score from a predefined list. In the experiments in which the number of the positive training cases is less than 5, we used a similar strategy as above but we divided the positive training set and the negative training set into 2 folds, and we did 2-fold CV on the newly formed set M to choose γ. In the experiments, the free parameters C for SVM and the free parameter γ are chose using Cross Validation as discussed above. Before training SVM, the kernel was normalized using $K(i,j) \leftarrow \frac{K(i,j)}{\sqrt{K(i,i)K(j,j)}}$.

Table 2 gives the mean ROC_{50} scores on different protein classes in several classes corresponding to the original kernels and the modified kernels. From Table 2, we see that:

modified kernels using label information gave better performance than the original kernels.

To determine whether the improvement given by the modified kernels is statistically significant, we performed a Wilcoxon Matched-Pairs Signed-Ranks Test on the differences. The resulting p-value for the improvement over the Mismatch+homologs [PSI-BLAST] kernel is $2.19e-04$, and the p-value for the improvement over the Profile+homologs [PSI-BLAST] kernel is 0.0162.

To show our algorithm improves the original kernels in more detail, in Figure 2 and Figure 3, we respectively plot a block sub-matrix of the test part of the normalized original Profile + [PSIBLAST] and of the normalized improved Profile +[PSIBLAST] matrix on family 2.28.1.1 (Legume lectins). In the two figures, the first three rows correspond to all the positive test sequences in the test set, and the remainder rows correspond to some randomly selected negative test sequences. The first nine columns correspond to some randomly selected positive training sequences, and the last column corresponds to a randomly selected negative training sequence. The whiter the blobs in the figures, the larger the corresponding similarity scores. Comparing Figure 2 to Figure 3, we can see that the similarity scores between the positive test data and the positive training data in the improved kernel is increased (the block on the upper left corner becomes whiter in the improved kernel matrix).

6 Discussion

We described an approach to modify kernels using label information of training data based on SVD and a linear mapping. The modified kernel is more discriminative than the original kernel. We also showed that, unlike Kernel Alignment with SDP, the test part of the modified kernel can be calculated very efficiently in practice. We tested the performance of the modified kernel by performing handwritten digit classification and detecting protein remote homology. Experimental results show that the improvement given by the new kernel is significant, although one more free parameter γ is introduced.

We believe that the modified kernel will not overfit the training data, because the label information is only used to modify the training part of kernel matrix and the degree of the modification is controlled by CV. The experimental results in the paper show that the generalization is good. The approach discussed in the paper is general and can be readily applied to many problems. In the future work, we plan to apply the approach discussed to gene function prediction problems.

Acknowledgment

This project was funded by a start-up fund from University of Toronto to Zhaolei Zhang, an NSERC grant to Anthony Bonner, and a grant from Genome Canada through the Ontario Genomics Institute.

References

[1] Jaakkola, T., Diekhans, M., and Haussler, D.: A discriminative framework for detecting remote protein homologies. Journal of Computational Biology. **7** (2000) Numbers 1/2, 95-114.

[2] Kuang, R., Ie, E., Wang, K., Wang, K., Siddiqi, M., Freund, Y., and Leslie C.: Profile-based String Kernels for Remote Homology Detection and Motif Extraction. Journal of Bioinformatics and Computational Biology. **3** (2005) No. 3 527-550.

[3] Krogh, A., Brown, M., Mian, I., Sjolander, K., and Haussler, D.: Hidden markov models in computational biology: Applications to protein modeling. Journal of Molecular Biology. **235** (1994) 1501-1531.

[4] Baldi, P., Chauvin, Y., Hunkapiller, T., and McClure, M.A.: Hidden markov models of biological primary sequence information. PNAS, **91(3)** (1994) 1059-1063.

[5] Murzin A. G., Brenner S. E., Hubbard T., Chothia C.: SCOP: a structural classification of proteins database for the investigation of sequences and structures. J. Mol. Biol. **247** (1995) 536-540.

[6] Weston, J., Leslie, C., Ie, E., Zhou, D., Elisseeff, A. and Noble, W.S.: Semi-Supervised Protein Classification using Cluster Kernels. Bioinformatics. **21** (2005) 3241-3247.

[7] Zhu, X., Kandola, J., Ghahramani, Z., and Lafferty, J.: Nonparametric Transforms of Graph Kernels for Semi-Supervised Learning. Advances in Neural Information Processing Systems. **17** (2005).

[8] Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L. and Jordan, M.: Learning the kernel matrix with semidefinite programming. Journal of Machine Learning Research. **5** (2004) 27-72.

[9] Kwok, T., Tsang, I.W.: Learning with idealized kernels. Proceedings of the Twentieth International Conference on Machine Learning (ICML), (2003) pp. 400-407.