

Abstract

Microarray technique measures gene expression levels under various conditions, simultaneously. Microarray data are successfully analyzed by kernel methods for a variety of applications. A major drawback of microarray is technically error prone. To gain accurate analysis, we propose a method which produces a noise-tolerant kernel matrix. First of all, we devise a new distance function for microarray data. The distance function is robust to noise, but not metric. Although we need a kernel matrix to apply kernel methods, yet construction of a kernel matrix from the non-metric distances is not an easy task. Our algorithm for building a kernel matrix is based on a maximum-entropy method using constraints derived from the distances. Promising results are shown in classification and network inference.

kernel methods, microarray data, partial distance, maximum entropy method, network inference

Kernel Analysis for Noisy Microarray Data

Tsuyoshi Kato^{*†}, Wataru Fujibuchi[†] and Kiyoshi Asai^{*†}

[†] AIST Computational Biology Research Center
2-43, Aomi, Koto-ku, Tokyo, 135-0064 Japan

^{*} Graduate School of Frontier Sciences, The University of Tokyo
5-1-5, Kashiwanoha, Kashiwa, 277-8562, Japan
{kato-tsuyoshi, fujibuchi-wataru, asai-cbrc}@aist.go.jp

June 12, 2006

1 Introduction

Microarray technique (e.g. [15]) has been proving immensely valuable to cell biologists, scientists and drug researchers, by being able to track tens of thousands of molecular reactions in parallel. Microarray technology aims at assessing the transcript abundances (measured in terms of fluorescence intensity) of thousands of genes in response to different experimental conditions or in different tissue samples.

Microarray data are usually given as a matrix; the row and column specify to which gene and which cell (or experiment) a particular piece of data corresponds, respectively. A matrix can be regarded as a set of feature vectors with fixed length. The data representation is assumed in many statistical methods. For example, if we identify the type of cells, each column vector of the matrix is a feature vector. If we predict the biological function of genes, each row vector is a feature vector. As a consequence, the representation of microarray data facilitates the use of well established statistical techniques for a variety of analyses.

Among a number of statistical techniques, kernel methods have been proven to be a class of the most prominent learning algorithms by lots of studies. Particularly, support vector machines based on kernel methods are well-known and applied to many classification problems (e.g. [4]). Additionally, kernel PCA [16], kernel hierarchical clustering [18], SDP-SVM [14], and supervised network inference methods [10, 25] are also instances of kernel methods that offer analyses of microarray data. Nowadays kernel methods are a primary tool for analysis of microarray data.

Kernels are necessary for kernel methods [21]. Basically, kernels are similarities among feature vectors. For example, one can choose a kernel called linear kernel that is the standard inner-product in input space. Alternatively, one can also choose a kernel which is an inner-product among feature vectors after implicitly mapped

into a kernel Hilbert space. For example, RBF kernel and polynomial kernel [21] are such kernels. Since kernels are core components for measuring similarities, the choice of kernel is crucial to obtain accurate analysis.

A particular drawback of the microarray techniques is that running microarray experiments can be technically rather error prone. Microarray devices may contain dust and scratches, which may lead to the fails of spotting and hybridization process for some spots that represent gene expression levels. Therefore the microarray data frequently contain noisy values that may seriously disturb the subsequent statistical analysis. Nevertheless, most of analysis methods have been developed without taking into account the serious influence of noise so far.

One particular approach for handling noisy data is to reduce noise (de-noise) before data analysis. Typical de-noising methods are based on projection onto the principal subspace [12, 20]. Namely, each feature vector is projected onto the principal subspace which can be obtained by singular-value decomposition. For use of kernel methods, the kernel values are computed from the projected feature vectors. If the true distribution of data without noise were known, we could obtain the exact principal subspace of the true distribution and hereby this approach would work well. Typically, however, we know neither the true distribution nor the true principal subspace in advance. Hence we have to resort the contaminated data themselves to obtain the principal subspace. Thereby the principal subspace can still be contaminated, and the resulting projections are not well-denoised frequently.

In this paper, we present a new approach for kernel construction for such noisy microarray data. For reducing the influence of noise on microarray data, our approach defines a new distance function between feature vectors. Whereas the squared Euclidean distance function is defined by the sum of squared differences for all dimensions, the new distance function is for a subset of dimensions, and the subset are adaptively chosen. The new distance function tends to exclude noisy elements, so it is robust to noise. Although we need a kernel matrix to apply kernel methods, yet construction of a kernel matrix from the non-metric distances is not an easy task. Our algorithm for building a kernel matrix is based on a maximum-entropy method [11, 24]. The algorithm builds a kernel matrix by maximizing its entropy subject to the constraints derived from the distances. Even if the distances are non-metric, the algorithm works in a theoretically well-found framework.

In terms of kernel construction, there are similar attempts which embed examples into some vector space using distances. Presumably, the most popular attempts are the multi-dimensional scaling [23] and the locally linear embedding [19]. These two methods transfer the example data into a low dimensional Euclidean space (typically two-dimensional). Weinberger et al. [27] have presented a method for learning a kernel matrix so that examples are mapped into a (typically low-dimensional) kernel Hilbert space instead of a low dimensional Euclidean space. All of them have been developed with a common purpose which is visualization. Therefore, all the examples are projected into a common low-dimensional space even if they belong to different classes. Mostly those projections are preferable for visualization but not good for predictive purposes such as classification.

Besides our algorithm and the above mentioned methods, there are other meth-

ods based on distances. For classification, the k -nearest neighbor classifier is presumably the most well-known method based on distances. For clustering, the single-linkage method is widely known. The benefit of our algorithm is that kernel matrices are obtained. Our algorithm thereby provides a device to combine any method utilizing kernels with distances. Therefore the applications are not limited to classification or clustering. For example, the kernel matrix obtained from our algorithm can selectively be integrated with other types of data using SDP-SVM [14] or support kernel machine [1]. Supervised network inference methods [10, 25] require kernel matrices among nodes. Those kernel methods can work properly only when a supplied kernel satisfies positive-definiteness. Thence transformation into valid and appropriate kernel matrices is necessary for operation of those methods, and new kernels built on direct distance data allow the range of applications to be expanded.

This paper is organized as follows: The next section recalls some basic preliminaries for kernel methods. In section 3 we define the new distance function and describe the algorithm to construct a kernel matrix from the distances. Section 4 and 5 show the experimental results on classification and network inference, respectively. The last section concludes our paper.

2 Preliminaries

Kernel methods work by embedding the data into a kernel Hilbert space \mathcal{F} . The embedding is performed implicitly, by defining the inner product between each pair of examples rather than by giving their actual values of vectors [21]. Given a set of input examples $\mathbf{x}_i \in \mathcal{X}$, ($i = 1, \dots, N$) and an embedding space \mathcal{F} , we consider a mapping function $\Phi(\cdot) : \mathcal{X} \rightarrow \mathcal{F}$. Given two examples $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, the function $k(\cdot, \cdot)$ giving the inner product between their images, say $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}_j)$ in the space \mathcal{F} , is called a kernel function. A kernel matrix $K \in \mathfrak{R}^{N \times N}$ is a symmetric positive definite matrix of which elements are the values of the kernel function for $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathcal{X}$. Conversely, every symmetric positive definite matrix is a kernel matrix.

In this paper we assume a transductive setting [9] where we are given both labeled examples and test examples in advance in a learning stage. In a transductive setting, we do not have to know the kernel function $k(\cdot, \cdot)$ nor the implicitly defined mapping function Φ , nor the actual values of the images $\Phi(\mathbf{x}) \in \mathcal{F}$, if we have a kernel matrix. For example, the SVM score of j -th example can be written as $f_j = \sum_{i=1}^{\ell} \alpha_i y_i K_{ij} + b$, where the first $\ell (< N)$ examples are labeled by $y_i \in \{\pm 1\}$. α_i and b are the parameters determined by the SVM learning algorithm [21]. Notice that the scores are expressed by only kernel values and do not explicitly include any actual vectors in \mathcal{F} . For a given learning task, an important point is that of choosing a kernel, which corresponds to choosing a kernel matrix.

3 Method

Our method builds a kernel matrix from newly devised distances among examples. After we define the distance function, we describe the algorithm for building the kernel matrix.

3.1 Partial distance

Euclidean distance function is the most popular distance function among feature vectors. The squared Euclidean distance function is defined by the sum of the squared differences on each of the dimensions. For $\mathbf{x}, \mathbf{y} \in \mathfrak{R}^d$, the squared Euclidean distance function is expressed as

$$D_{\text{std}}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^d (x_i - y_i)^2 \quad (1)$$

Here we consider summation of the squared differences for a subset of elements:

$$\sum_{i \in \mathcal{I}} (x_i - y_i)^2 \quad (2)$$

where \mathcal{I} is an index set. If \mathcal{I} is a full set of indices, i.e. $\mathcal{I} = \{1, \dots, d\}$, this distance function is equivalent to the standard squared Euclidean distance function.

How should we choose the subset of indices \mathcal{I} ? Consider the case where a part of elements are contaminated. Those elements should be excluded from the index set \mathcal{I} . As described later, we need the values of the distance function for only the pairs such that an example in each pair is near to the partner. Hence the distances for such pairs should be small. From that consideration, we devise the new distance function which incorporates the d_p smallest element-wise differences and ignores the $(d - d_p)$ remaining differences, where d_p is chosen manually in advance. The new distance function is defined by

$$D_p(\mathbf{x}, \mathbf{y}) = \min_{\mathcal{I} \text{ s.t. } |\mathcal{I}| = d_p} \sum_{i \in \mathcal{I}} (x_i - y_i)^2, \quad (3)$$

where $|\mathcal{I}|$ denotes the cardinality of \mathcal{I} . We refer the distance function defined in Eq. (3) to the *partial distance* function. The dimensions including noise in one-side or both-sides are likely to be outliers. This distance function can effectively exclude the dimensions being outliers by selecting d_p dimensions which are of the minimum differences.

Similar distance functions are also used in motion estimation in video coding [3]. In their method, the differences for a fixed subset of pixels are summed up for computing distances among blocks. It is for speedup, not for ignoring noise. Therefore their purpose is quietly different from ours, whereas their definition is similar.

The relation between $D_{\text{std}}(\cdot, \cdot)$ and $D_p(\cdot, \cdot)$ resembles the relation between the cost function of Hartley and Schaffalitzky [8] and the cost function of Ke and Kande [13] in the field of computer vision. The squared Euclidean distance function

$D_{\text{std}}(\mathbf{x}, \mathbf{y})$ is the squared L_2 -norm of the vector $(\mathbf{x} - \mathbf{y})$. Hartley and Schaffalitzky [8] use L_∞ -norm instead of L_2 -norm for the cost function of geometric reconstruction. The reason for replacement of L_2 -norm with L_∞ -norm is for facilitation of numerical optimization. However, as is also pointed out in [8], L_∞ -norm is sensitive to outliers. Ke and Kanade [13] propose the point-wise d_p -th smallest projection error as the cost function. Their cost function is robust to outliers, yet optimization of the cost function is still efficient. In our case, the distance function is not directly used as the objective function for any optimization problems. Therefore we do not need to replace L_2 -norm with L_∞ -norm. Hence we have devised the distance function in Eq. (3) which is more naturally derived from the squared Euclidean distance function.

For analysis of microarray data using kernel methods, we need a kernel matrix. We consider building the kernel matrix from the partial distances. RBF kernel is a well-known kernel which is computed from distances among examples. Given a distance between \mathbf{x} and \mathbf{y} , say $D(\mathbf{x}, \mathbf{y})$, RBF kernel is defined by

$$K_{\text{rbf}}(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{D(\mathbf{x}, \mathbf{y})}{2\sigma^2}\right),$$

where σ is a constant parameter. Usually the squared Euclidean distance function is used for $D(\cdot, \cdot)$. Generally, the RBF kernel is valid if and only if $D(\cdot, \cdot)$ is negative definite [21]. The squared Euclidean distance function $D_{\text{std}}(\cdot, \cdot)$ is negative definite, but the partial distance function $D_p(\cdot, \cdot)$ is not. Hence we have to employ another approach to convert the partial distances to a kernel matrix.

3.2 Kernel Construction

We here describe an algorithm for construction of a kernel matrix from the partial distances. The algorithm is based on the maximum entropy method [11, 24] which allows use of non-metric distances for kernel methods by producing a kernel matrix.

Speaking again, kernels must be the inner-product on a kernel Hilbert space. We wish to obtain a kernel matrix in which the points in the kernel Hilbert spaces \mathcal{F} keeping the partial distances among those points. Denote the examples by $\mathbf{x}_i \in \mathcal{X}$ ($i = 1, \dots, N$) and the mapping function by $\Phi : \mathcal{X} \mapsto \mathcal{F}$. We then give the following constraints to the images:

$$\|\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)\|^2 \leq cD_p(\mathbf{x}_i, \mathbf{x}_j) \quad (4)$$

where c is a constant parameter. Those constraints are given not to all the pairs between examples, but to the pairs each of which has a small partial distance. For determine such pairs, we couple each example with k nearest neighbors in our simulations. Denote the set of index pairs to be given the constraints by $\mathcal{J} \equiv \{(i_k, j_k)\}_{k=1}^M$. As described in the previous section, kernel methods work on a kernel matrix with elements $K_{ij} = \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j)$. Using the elements of a kernel matrix, the squared Euclidean distance between $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}_j)$ is described as $\|\Phi(\mathbf{x}_i) -$

$\Phi(\mathbf{x}_j)\|^2 = K_{ii} + K_{jj} - K_{ij} - K_{ji}$. For the sake of simple notation, we define U_k ($k = 1, \dots, M$) by

$$U_k = E_{i_k i_k} + E_{j_k j_k} - E_{i_k j_k} - E_{j_k i_k} - cD_p(\mathbf{x}_{i_k}, \mathbf{x}_{j_k})I$$

where E_{ij} denotes a matrix in which (i, j) element is one and all the others are zero. For any kernel matrix K such that $\text{tr} K = 1$, the distance constraints can be rewritten as $\text{tr}(U_k K) \leq 0$ for $k = 1, \dots, M$. Under the constraints, we wish to construct a kernel matrix with least irrelevant information. Such a kernel matrix $K \in \Re^{N \times N}$ is obtained by maximization of the von Neumann entropy [17] defined by $-\text{tr}(K \log K)$, where \log takes the matrix logarithm.

If given partial distances actually violate the triangle inequalities, the optimization problem may become infeasible. For keeping the optimization problem feasible, we relax the problem like soft-margin support vector machine as follows:

$$\begin{aligned} \min \quad & \text{tr}(K \log K) + \lambda \|\boldsymbol{\xi}\|_1, \\ \text{subj. to} \quad & \text{tr}(U_k K) \leq \xi_k, \quad k = 1, \dots, M, \\ & \boldsymbol{\xi} \geq \mathbf{0}, \quad \text{tr} K = 1, \\ \text{w.r.t.} \quad & K, \boldsymbol{\xi} = [\xi_1, \dots, \xi_M]^\top, \end{aligned} \tag{5}$$

where λ is constant. Since the entropy function is convex, the optimization function is convex. Inasmuch as all the constraints are linear, the feasible region is a convex set. Consequently, the optimization problem does not have any local minima, and we can always attain to the optimal solution.

When determining the constant parameter c , we should take account of the constraint: the trace of K must be one. Due to that constraint, the average of squared norms $\|\Phi(\mathbf{x}_i)\|^2$ becomes $1/N$. Hence a suitable heuristic is to set $c \propto 1/N$. In our simulations, we set that constant parameter by $c = \frac{1}{4ND_{\text{avg}}}$, where D_{avg} is the average of the partial distances between the pairs with the constraints: $D_{\text{avg}} = \frac{1}{M} \sum_{k=1}^M D_p(\mathbf{x}_{i_k}, \mathbf{x}_{j_k})$.

Optimization algorithm There is a steepest descent algorithm for a general problem with this type of the objective function and linear constraints [24]. We employ that algorithm for solving our optimization problem. Here we describe it briefly as the following. The algorithm solves the dual problem instead of the primal problem given in Eq. (5). The dual problem is described by:

$$\begin{aligned} \max \quad & J(\boldsymbol{\alpha}) = -\log \text{tr}(\exp(-\mathcal{U}\boldsymbol{\alpha})). \\ \text{subj. to} \quad & \mathbf{0} \leq \boldsymbol{\alpha} \leq \lambda \mathbf{1}, \\ \text{w.r.t.} \quad & \boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_M]^\top, \end{aligned} \tag{6}$$

where $\boldsymbol{\alpha}$ are a dual variable vector, and the operator \mathcal{U} performs $\mathcal{U}\boldsymbol{\alpha} = \sum_{k=1}^M \alpha_k U_k$. $\mathbf{1}$ denotes a column vector in which all elements are one. For optimization, the steepest descent method is used. The derivatives of the dual function is given by

$$\frac{\partial J}{\partial \alpha_k} = \frac{\text{tr}(U_k \exp(-\mathcal{U}\boldsymbol{\alpha}))}{\text{tr}(\exp(-\mathcal{U}\boldsymbol{\alpha}))}. \tag{7}$$

When the values of some dual variables violate the constraints in (6), they are forced to be back into the feasible region. Since the optimization problem is convex, the optimal solution can always be attained from any initial values. In our simulations, we put zero to the initial values of the dual variables. Once we get the dual optimal, we can recover the primal optimal solution as follows:

$$K = \frac{\exp(-\mathcal{U}\boldsymbol{\alpha})}{\text{tr}(\exp(-\mathcal{U}\boldsymbol{\alpha}))}. \quad (8)$$

Let us discuss the time complexity of the optimization algorithm. U_k are sparse. Denote the number of non-zeros in each matrix by N_{nz} . If we use a special data structure for sparse matrices (e.g. Harwell-Boeing sparse matrix storage format [5]), addition $S = -\mathcal{U}\boldsymbol{\alpha} \in \mathfrak{R}^{N \times N}$ takes $O(N_{\text{nz}}M)$. Computation of matrix exponential of S takes $O(N^3)$. The trace of the product between U_k and $S' = \exp(S)$, say $\text{tr}(U_k S')$, requires the computational time $O(N_{\text{nz}})$, because that trace can be rewritten as the inner product of vectors with N_{nz} elements. Computing the trace of S' also takes $O(N)$. The number of non-zeros of U_k is $N + 2$. Hence, after obtaining S' , we need $O(N)$ for computation of the gradient Eq. (7) for each element in a dual vector. Recovery of the kernel matrix takes $O(NM + N^3 + N)$. If we assume $M < N^2$, the total computational time is therefore $O(T_{\text{iter}}N^3)$ where the number of iterations is T_{iter} .

4 Simulations of Classification

In this section we experimentally compare our kernel with existing methods using a classification problem. Typical conventional de-noising methods project each example to the principal subspace. For experimental comparison, we actually implement the de-noising method. The method obtains the principal subspace by singular value decomposition (SVD) of the example matrix $X = [\mathbf{x}_1, \dots, \mathbf{x}_N]$ and de-noises each of example by projection onto the principal subspace. Then the kernel matrix among the projections is computed. For getting the kernel matrix, we use the linear kernel and the RBF kernel. We refer the linear kernel and the RBF-kernel of the de-noised data to SVD-linear kernel and SVD-RBF kernel, respectively. The existing kernels of the raw data are here called linear kernel and RBF kernel, simply. Our kernel built using the partial distance function is referred to partial distance kernel (PD kernel), and the kernel built using $D_{\text{std}}(\cdot, \cdot)$ to standard distance kernel (SD kernel).

It is widely recognized that an accurate diagnosis is important to the design of an adequate treatment protocol and ultimately to the survival of the patient. We here carry out the simulations for diagnosis of cell types in order to examine the performance of PD kernel. In the simulations, we use the microarray dataset containing close but two different cell types called AML and ALL [6]. To collect this dataset, bone marrow or blood samples were taken from 72 patients including 47 with acute myeloid leukemia (AML) and 25 with acute lymphoblastic leukemia (ALL). Here the task is here to classify each cell to one of the two classes, AML and ALL. We extract 100 genes from the dataset by the software RankGene [22].

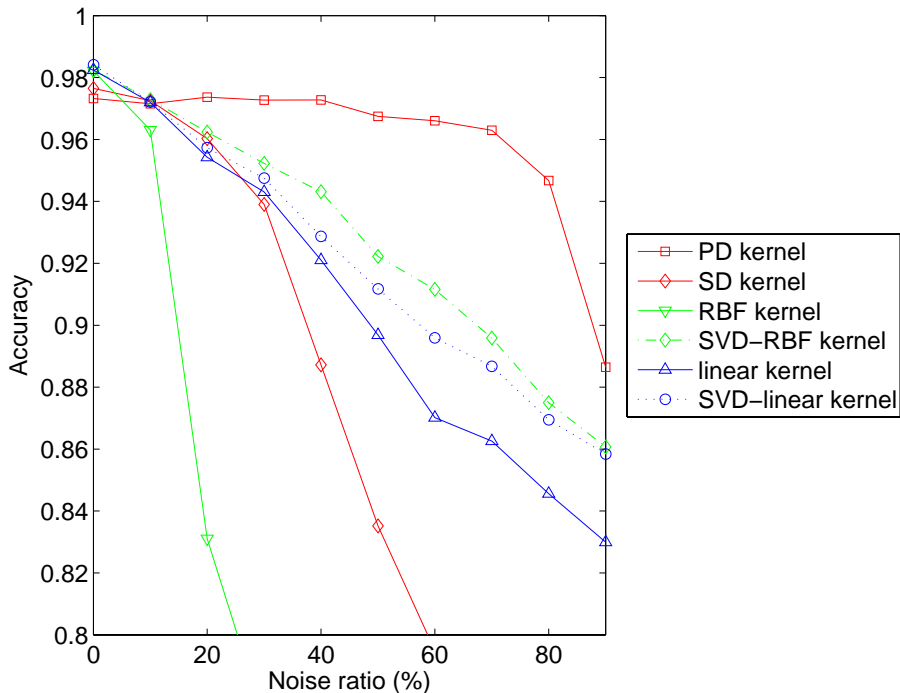


Figure 1: Accuracies for classification.

To create noisy data, we added the Gaussian noise with standard deviation two to various percentages of the data. Then we compute the partial distances, find five nearest neighbors for each cell, and give the distance constraints to those pairs to build the kernel matrix. Independently we pick 23 AML cells and 12 ALL cells randomly and give them class labels. Then the classification is done for 37 unlabeled cells using SVM with the regularization parameter $C = 1$. We repeated this 100 times and compute each of accuracy. For reducing the influence of the choice of random noise, we furthermore repeated the overall procedure ten times and take the average of the 1,000 accuracies.

In all the experiments, the constant parameters of PD kernel are set as $d_p = \lfloor d/2 \rfloor = 50$, $\lambda = 100$. For the existing kernels, we varied the dimension of the principal subspace with $d_{pca} = 1, 2, 3, 4, 5, 7, 10, 20, \dots, 100$. The parameter of RBF kernel is tested with $\sigma = 10^{0.5}, 10^1, 10^{1.5}, 10^2, 10^{2.5}, 10^3$. We report the accuracies on the parameter set yielding the maximum accuracy in each noise ratio.

The experimental results are depicted in Figure 1. For the noise ratio between 20% and 90%, PD kernel achieves significant improvement. Compared the kernels using SVD with the kernels with raw data, SVD improves the accuracies. However the performances do not amount to the accuracy of PD kernel. At no noise ratio the existing kernels are slightly superior to PD kernel, but the differences are comparative. Remarkably the classification performances of PD kernel are almost kept until 70% noise ratio. Accordingly the robustness of PD kernel is shown by the classification experiments.

5 Simulations of Network Inference

In this section we examine the performance of PD kernel in the setting of network inference. A goal of molecular biology is to understand the underlying mechanism of cellular processes. For extraction of meaningful insights about these processes, networks of proteins are handy and useful, wherefore inference of the network is a central issue of computational biology [10, 25, 28, 19]. Although several methods have been developed in different assumptions [10, 25], we employ the most basic method for network inference. The method is to compute the kernel matrix among nodes and then pick up the elements higher than a threshold for estimating the adjacency matrix of a network [28].

We test PD kernel on the protein interaction network related to carbohydrate metabolism provided by von Mering et al. [26]. Each edge is rated with a confidence level: high, middle, or low. We only used high confidence edges because they are supported by multiple experiments. By removal of proteins without any edges, we obtained a network of 49 proteins and 51 edges.

The kernel matrix is computed using microarray data. The microarray data consist of 330 experiments collected from Stanford microarray database [2]. Notice that a cell is an example of kernel methods in the simulations in the previous section, whereas a gene corresponds to an example here. We report the results under the assumption that the microarray data are noisy. The procedure for simulating contamination is same as the previous section. Using the procedure we generate 50 noise patterns artificially for a given noise ratio. We measure the prediction performance using the ROC score [7]. The ROC score is the area under the receiver operating characteristic (ROC) curve, which plots the ratio of true positives against the ratio of false positives for different possible thresholds [7]. We report the average of 50 ROC scores, each of which is obtained from the microarray data contaminated by one of 50 noise patterns. The constant parameters of PD kernel are set as $d_p = \lfloor d/2 \rfloor = \lfloor 330/2 \rfloor = 165$ and $\lambda = 100$. We also test the existing kernels used in the previous section. We determine the constant parameters of the existing kernels by the grid search over the values: $d_{pca} = 1, 2, 3, 4, 5, 7, 10, 20, 30, 60, 90, 120, 150, 210, 270, 330$, $\sigma = 10^{-0.5}, 10^0, 10^{0.5}, 10^1, 10^{1.5}, 10^2$; the combination of values yielding the best ROC score is chosen for each noise ratio.

The experimental results are shown in Figure 2. The PD kernel exhibits the compelling robustness. Even in the case without artificial noise, the performance of PD kernel is significantly superior to the existing kernels. For linear kernels, the effect of de-noising by SVD is observed even in the case without artificial noise. This implies that the original microarray data have already been contaminated by seriously large noise. Consequently the results reveal that PD kernel is considerably robust to noise in the network inference setting as well as classification setting.

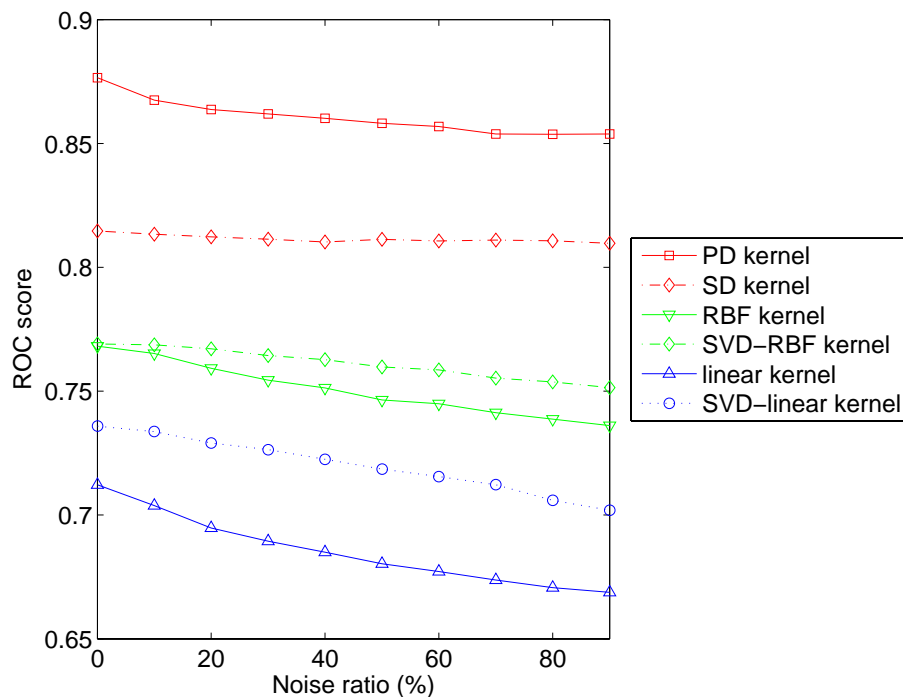


Figure 2: ROC scores for network inference.

6 Conclusion

We have presented a computational method to obtain a noise-tolerant kernel matrix and shown the encouraging experimental results in the setting of classification and network inference. Although our kernel is designed for the purpose of noisy microarray data analysis, it can be applied directly for any vectorial data. Future work will explore other applications of our kernel.

References

- [1] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.
- [2] C. A. Ball, I. A. Awad, J. Demeter, J. Gollub, J. M. Hebert, T. Hernandez-Boussard, H. Jin, J. C. Matese, M. Nitzberg, F. Wymore, Z. K. Zachariah, P. O. Brown, and G. Sherlock. The stanford microarray database accommodates additional microarray platforms and data formats. *Nucleic Acids Res. Database issue*, 33:D580–582, Jan 2005.
- [3] C.-D. Bei and R. M. Gray. An improvement of the minimum distortion encoding algorithm for vector quantization. *IEEE Transactions on Communications*, 33(10):1132–1133, Oct. 1985.

- [4] F. Chu and L. Wang. Applications of support vector machines to cancer classification with microarray data. *Int. J. Neural Syst.*, 15(6):475–484, Dec. 2005.
- [5] I. S. Duff, R. G. Grimes, and J. G. Lewis. Sparse matrix test problems. *ACM Trans. Math. Software*, 15:1–14, 1989.
- [6] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Lohand J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, Oct. 1999.
- [7] M. Gribskov and N. L. Robinson. Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching. *Computers and Chemistry*, 20(1):25–33, 1996.
- [8] R. Hartley and F. Schaffalitzky. l_∞ minimization in geometric reconstruction problems. In *CVPR*, 2004.
- [9] T. Joachims. Transductive inference for text classification using support vector machines. In *Proc. 16th International Conference on Machine Learning*, pages 200–209, San Francisco, CA, 1999. Morgan Kaufmann.
- [10] T. Kato, K. Tsuda, and K. Asai. Selective integration of multiple biological data for supervised network inference. *Bioinformatics*, 21:2488–2495, Feb 2005.
- [11] Tsuyoshi Kato, Wataru Fujibuchi, and Kiyoshi Asai. Learning kernels from distance constraints. *IPSJ Digital Courier*, 2006. to appear.
- [12] Tsuyoshi Kato, Yukio Murata, Koh Miura, Kiyoshi Asai, Paul B. Horton, Koji Tsuda, and Wataru Fujibuchi. Network-based de-noising improves prediction from microarray data. *BMC Bioinformatics*, 7(Suppl. 1):S4, March 2006.
- [13] Q. Ke and T. Kanade. Quasiconvex optimization for robust geometric reconstruction. In *ICCV*, 2005.
- [14] G. R. G. Lanckriet, T. De Bie, Nello Cristianini, M.I. Jordan, and W.S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20:2626–2635, 2004.
- [15] Mei-Ling Ting Lee. *Analysis of Microarray Gene Expression Data (Trends in Logic S.)*. Kluwer Academic Pub, March 2004.
- [16] Z. Liu, D. Chen, and H. Bensmail. Gene expression data classification with kernel principal component analysis. *J. Biomed. Biotechnol.*, 2:155–159, June 2005.
- [17] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge Univ Press, 2000.

- [18] J. Qin, D. P. Lewis, and W. S. Noble. Kernel hierarchical gene clustering from microarray expression data. *Bioinformatics*, 19(16):2097–2104, Nov. 2003.
- [19] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.
- [20] G. Sanguinetti, M. Milo, M. Rattray, and N. D. Lawrence. Accounting for probe-level noise in principal component analysis of microarray data. *Bioinformatics*, 21(19):3748–3754, 2005.
- [21] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
- [22] Y. Su, T. M. Murali, V. Pavlovic, M. Schaffer, and S. Kasif. RankGene: identification of diagnostic genes based on expression data. *Bioinformatics*, 19:1578–1579, Aug 2003.
- [23] Michael A. A. Cox Trevor F. Cox. *Multidimensional Scaling, Second Edition*. Chapman & Hall, Sept. 2000.
- [24] K. Tsuda and W.S. Noble. Learning kernels from biological networks by maximizing entropy. *Bioinformatics*, 20(Suppl. 1):i326–i333, 2004.
- [25] J.-P. Vert and Y. Yamanishi. Supervised graph inference. In Lawrence K. Saul, Yair Weiss, and Lon Bottou, editors, *Advances in Neural Information Processing Systems 17*, Cambridge, MA, 2005. MIT Press.
- [26] C. von Mering, R. Krause, B. Snel, M. Cornell, S. G. Oliver, S. Fields, and P. Bork. Comparative assessment of large-scale data sets of protein-protein interactions. *Nature*, 417:399–403, 2002.
- [27] Kilian Q. Weinberger, Fei Sha, and Lawrence K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the Twenty First International Conference on Machine Learning (ICML 2004)*, July 2004.
- [28] Y. Yamanishi, J. P. Vert, and M. Kanehisa. Supervised enzyme network inference from the integration of genomic data and chemical information. *Bioinformatics*, 21 Suppl. 1:i468–i477, Jun 2005.