A Statistical Framework for Genomic Data Fusion in Yeast

Gert R. G. Lanckriet
Department of Electrical Engineering
and Computer Science
University of California, Berkeley

Nello Cristianini
Department of Stastistics
University of California, Davis

Minghua Deng
Department of Biological Sciences
University of Southern California

Michael Jordan
Division of Computer Science
Department of Statistics
University of California, Berkeley

William Stafford Noble
Department of Genome Sciences
University of Washington

September 10, 2003

Abstract

During the past decade, the new focus on genomics has highlighted a particular challenge: to integrate the different views of the genome that are provided by various types of experimental data. We describe a computational framework for integrating and drawing inferences from a collection of genome-wide measurements. Each data set is represented via a kernel function, which defines generalized similarity relationships between pairs of entities, such as genes or proteins. The kernel representation is both flexible and efficient, and can be applied to many different types of data. Furthermore, kernel functions derived from different types of data can be combined in a straightforward and optimal fashion. The problem of optimal kernel combination can be formulated as a convex optimization problem that can be solved with semidefinite programming techniques. Computational experiments performed using yeast genome-wide data sets, including amino acid sequences, hydropathy profiles, gene expression data and known protein-protein interactions, demonstrate the utility of this approach.

1 Introduction

The recent availability of multiple types of genome-wide data provides biologists with complementary views of a single genome, and highlights the need for algorithms capable of unifying these views. In yeast, for example, for a given gene we typically know the protein it encodes, that protein's similarity to other proteins, its hydrophobicity profile, the mRNA expression levels associated with the given gene under hundreds of experimental conditions, the occurrences of known or inferred transcription factor binding sites in the upstream region of that gene, and the identities of many of the proteins that interact with the given gene's protein product. Each of these distinct data types provides one view of the molecular machinery of the cell. In the near future, research in bioinformatics will focus more and more heavily on methods of data fusion.

Different data sources are likely to contain distinct and thus partly independent information. Combining those complementary pieces of information can be expected to enhance the total information about the problem at hand. One problem with this approach, however, is that genomic data come in a wide variety of data formats: expression data are expressed as vectors or time series; protein sequence data as strings from a 20-symbol alphabet; gene sequences are strings from a different (4-symbol) alphabet; protein-protein interactions are best expressed as graphs, and so on.

This paper presents a computational and statistical framework for integrating heterogeneous descriptions of the same set of genes. The approach relies on the use of kernel-based statistical learning methods that have already proven to be very useful tools in bioinformatics [13]. These methods represent the data by means of a kernel function, which defines similarities between pairs of genes, proteins, etc. Such similarities can be quite complex relations, implicitly capturing aspects of the underlying biological machinery. When different data types are available, one can construct a series of kernel functions, each of them extracting a specific type of information from each of the available data types, thereby providing a partial description or view of the data. The advantage of the kernel representation is that it is both flexible and efficient, and can be applied to many different types of data, including sequence-strings, interaction graphs, expression profiles, and so on. Our goal is now to merge all information extracted by this series of kernels by finding their optimal combination, i.e., the combined kernel that best represents all of the information available for a given statistical learning task. We solve the

mathematical problem of combining kernels using a convex optimization method known as semidefinite programming (SDP) [12, 16]. This SDP-based approach [10] yields a general methodology for combining many partial descriptions of data that is statistically sound, as well as computationally efficient and robust.

We begin by outlining the main ideas of the kernel approach to pattern analysis, providing examples of kernels defined on yeast genome-wide data sets. We then describe how these kernels can be integrated using SDP to provide a unified description. Finally, we describe a series of computational experiments that demonstrate the validity and power of the kernel approach to data fusion for membrane protein recognition and protein function prediction in yeast.

2 Kernel Methods

Kernel methods work by embedding data items (genes, proteins, etc.) into a vector space $\mathcal{F}$, called a *feature space*. A key characteristic of kernel methods is that the embedding in feature space is generally defined implicitly, by specifying an inner product for the feature space. Thus, for a pair of data items, x_1 and x_2, denoting their embeddings as $\Phi(x_1)$ and $\Phi(x_2)$, respectively, we specify the inner product of the embedded data, $\langle \Phi(x_1), \Phi(x_2) \rangle$, via a *kernel function* $K(x_1, x_2)$. Any symmetric, positive semidefinite function is a valid kernel function, corresponding to an inner product in some feature space. Note that if all we require are inner products, then we do not need to have an explicit representation of the mapping Φ, nor do we even need to know the nature of the feature space. It suffices to be able to evaluate the kernel function.

Evaluating the kernel on all pairs of data points yields a symmetric, positive semidefinite matrix known as the *kernel matrix* or the *Gram matrix*. Intuitively, a kernel matrix can be regarded as a matrix of generalized similarity measures among the data points. The first stage of processing in a kernel method is to reduce the data by computing this matrix.

The reduction to a kernel matrix reflects the fact that kernel methods are generally based on linear statistical procedures in feature space. In particular, the classification algorithm that we use in this paper—known as a *support vector machine* (SVM) [4]—forms a linear discriminant boundary in feature space. Consider a data set consisting of n pairs (x_i, y_i), where x_i is the ith data item (e.g., a protein), and $y_i \in \{-1, 1\}$ is a label (e.g., membrane or non-membrane). Compute the $n \times n$ kernel matrix whose (i, j)th entry is $K(x_i, x_j)$. Given this matrix, and given the labels y_i, we can throw away the original data; the problem of fitting the SVM to data reduces to an optimization procedure that is based entirely on the kernel matrix and the labels.

Different kernel functions correspond to different embeddings of the data and thus can be viewed as capturing different notions of similarity. For example, in a space derived from amino acid sequences, two genes that are close to one another will have protein products with very similar amino acid sequences. This amino acid space would be quite different from a space derived from microarray gene expression measurements, in which closeness would indicate similarity of the expression profiles of the genes. In general, a single type of data can be mapped into many different feature spaces. The choice of feature space is made implicitly via the choice of kernel function.

Microarray expression data is among the easiest types of genomic data to define a kernel for, because each gene is represented by a fixed-length vector of real-valued expression measurements. A straightforward means of comparing such profiles is a normalized scalar product, which is related to the Pearson correlation coefficient. Other commonly used kernels include polynomial versions of the scalar product, as well as a Gaussian kernel known as a radial basis function.

Protein sequences are variable length sequences of letters from a discrete alphabet. As such, computing a scalar product directly is infeasible. Numerous methods exist for computing kernels on sequences. Perhaps the most straightforward is to leverage existing sequence comparison algorithms, such as BLAST [1] or Smith-Waterman (SW) [15]. Because matrices of BLAST or Smith-Waterman scores are not necessarily positive semidefinite, they cannot be used as kernels directly. However, a pairwise comparison kernel can be derived from a matrix of BLAST or SW scores simply by applying a standard kernel function to all pairs of rows in the matrix. The resulting kernel has been demonstrated to provide powerful remote homology recognition performance [11].

Pairwise interactions among proteins can be represented as a binary matrix, in which rows and columns are proteins and a "1" represents an interaction. This matrix representation allows for a straightforward kernel: the scalar product simply counts the number of proteins that a given pair of proteins interact with. An alternative way to represent the same interaction data is to consider the proteins as nodes in a large graph. In this graph, two proteins are linked when they interact and otherwise not. Kondor and Lafferty [8] propose a general method for establishing similarities between the nodes of a graph, based on a random walk on the graph. This method efficiently accounts for all possible paths connecting two nodes, and for the lengths of those paths. Nodes that are connected by shorter paths or by many paths are considered more similar. The resulting *diffusion kernel* can be applied directly to protein-protein interaction data.

3 Kernel Methods for Data Fusion

Each of the kernel functions described above produces, for the yeast genome, a square matrix in which each entry encodes a particular notion of similarity of one yeast protein to another. Implicitly, each matrix also defines an embedding of the proteins in a feature space. Thus, the kernel representation casts heterogeneous data—real-valued gene expression data, variable-length amino acid strings, and a graph of protein-protein interactions—into the common format of kernel matrices.

The kernel formalism also allows these various matrices to be combined. Basic algebraic operations such as addition, multiplication and exponentiation preserve the key property of positive semidefiniteness, and thus allow a simple but powerful algebra of kernels [2]. For example, given two kernel functions K_1 and K_2, inducing the embeddings $\Phi_1(x)$ and $\Phi_2(x)$, respectively, it is possible to define the kernel $K = K_1 + K_2$, inducing the embedding $\Phi(x) = [\Phi_1(x), \Phi_2(x)]$. Of even greater interest, we can consider parameterized combinations of kernels. In particular, given a set of kernels $\mathcal{K} = \{K_1, K_2, \ldots, K_m\}$, we can form the linear combination

$$K = \sum_{i=1}^{m} \mu_i K_i, \tag{1}$$

where the weights are constrained to be non-negative to assure positive semidefiniteness: $\mu_i \geq 0; i = 1, \ldots, m$. We consider this kind of kernel combination in this paper.

As we have discussed, fitting a kernel-based statistical classifier (such as the SVM) to data involves solving an optimization problem based on the kernel matrix and the labels. In particular, the SVM finds a linear discriminant in feature space that has maximal distance ("margin") between the members of the positive and negative classes. The algorithm for finding this optimal linear discriminant involves solving an optimization problem known as a *quadratic program*, a particular form of convex optimization problem for which efficient solutions are known [12].

The specific form of SVM that we use in this paper is the *1-norm soft margin support vector machine* [4, 14]. An SVM forms a linear discriminant boundary in the feature space $\mathcal{F}$: $f(x) = \mathbf{w}^T\Phi(x) + b$, where $\mathbf{w} \in \mathcal{F}$ and $b \in \mathbb{R}$. Given a labeled sample $S_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, a 1-norm soft margin SVM optimizes with respect to $\mathbf{w}$ and b so as to maximize the distance ("margin") between the positive and negative class, allowing misclassifications (therefore "soft margin"):

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\Phi(x_i) + b) \geq 1 - \xi_i, \;\; i = 1, \ldots, n \\ & \xi_i \geq 0, \;\; i = 1, \ldots, n \end{aligned} \tag{2}$$

where C is a regularization parameter, trading off error against margin. By considering the dual problem corresponding to Equation (2), one can prove [14] that the weight vector can be expressed as $\mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \Phi(x_i)$, where the support values α_i are solutions of the following dual *quadratic program* (QP):

$$\max_{\alpha} \quad 2\alpha^T\mathbf{e} - \alpha^T \text{diag}(\mathbf{y}) K \text{diag}(\mathbf{y})\alpha \;:\; C \geq \alpha \geq 0, \;\; \alpha^T\mathbf{y} = 0,$$

where $\mathbf{y} = (y_1, y_2, \ldots, y_n)^T$ and $\text{diag}(\mathbf{y})$ is a diagonal matrix with entries given by the elements of $\mathbf{y}$. An unlabeled data item x_{new} can subsequently be classified by computing the linear function

$$f(x_{new}) = \mathbf{w}^T\Phi(x_{new}) + b = \sum_{i=1}^{n}\alpha_i y_i K(x_i, x_{new}) + b.$$

If $f(x_{new})$ is positive, then we classify x_{new} as belonging to class $+1$; otherwise, we classify x_{new} as belonging to class -1.

In [10], we show that it is possible to extend such optimization problems not only to find optimal linear discriminant boundaries but also to find optimal values of the coefficients μ_i in Equation (1) for problems involving multiple kernels. In the case of the 1-norm soft margin SVM, this is achieved by minimizing the cost function in Equation (2) with respect to both the discriminant boundary and the μ_i. Again considering the Lagrangian dual problem, we can show that this problem of finding optimal μ_i and α_i reduces to a convex optimization problem known as a *semidefinite program (SDP)*:

$$\begin{aligned} \min_{\mu_i,t,\lambda,\nu,\delta} \quad & t \\ \text{subject to} \quad & \text{trace}\left(\sum_{i=1}^{m}\mu_i K_i\right) = c, \\ & \sum_{i=1}^{m}\mu_i K_i \succeq 0, \\ & \begin{pmatrix} \text{diag}(\mathbf{y})(\sum_{i=1}^{m}\mu_i K_i)\text{diag}(\mathbf{y}) & \mathbf{e} + \nu - \delta + \lambda\mathbf{y} \\ (\mathbf{e} + \nu - \delta + \lambda\mathbf{y})^T & t - 2C\delta^T\mathbf{e} \end{pmatrix} \succeq 0, \\ & \nu, \delta \geq 0, \end{aligned} \tag{3}$$

where c is a constant. SDP can be viewed as a generalization of linear programming, where scalar linear inequality constraints are replaced by more general linear matrix inequalities (LMIs): $F(\mathbf{x}) \succeq 0$, meaning that the matrix F has to be in the cone of positive

semidefinite matrices, as a function of the decision variables $\mathbf{x}$. Note that the first LMI constraint in Equation (3), $K = \sum_{i=1}^{m} \mu_i K_i \succeq 0$, emerges very naturally because the optimal kernel matrix must indeed come from the cone of positive semidefinite matrices. Linear programs and semidefinite programs are both instances of convex optimization problems, and both can be solved via efficient interior-point algorithms [16].

In this paper, the weights μ_i are constrained to be non-negative and the K_i are positive semidefinite and normalized ($[K_i]_{jj} = 1$) by construction; thus $K \succeq 0$ is automatically satisfied. In that case, we can show that the SDP in Equation (3) reduces to a *quadratically constrained quadratic program (QCQP)*, which is a special case of SDP that can be solved more efficiently:

$$\begin{aligned} \max_{\alpha,t} \quad & 2\alpha^T \mathbf{e} - ct \\ \text{subject to} \quad & t \geq \frac{1}{n}\alpha^T \text{diag}(\mathbf{y}) K_i \text{diag}(\mathbf{y})\alpha, \quad i = 1, \ldots, m \\ & \alpha^T \mathbf{y} = 0, \\ & C \geq \alpha \geq 0. \end{aligned} \tag{4}$$

Thus, by solving a QCQP, we are able to find an adaptive combination of kernel matrices—and thus an adaptive combination of heterogeneous information sources—that solves a classification problem. The output of our procedure is a set of weights μ_i and a discriminant function based on these weights. We obtain a classification decision that merges information encoded in the various kernel matrices, and we obtain weights μ_i that reflect the relative importance of these information sources.

4 Methods and Results

4.1 Membrane Protein Recognition

Membrane proteins anchor in one of various membranes in the cell and often serve important communicative functions. Generally, each membrane protein passes through the membrane several times. The transmembrane regions of the amino acid sequence are typically hydrophobic, whereas the non-membrane portions are hydrophilic. This specific hydrophobicity profile of the protein allows it to anchor in the cell membrane. Because the hydrophobicity profile of a membrane protein is critical to its function, this profile is better conserved in evolution than the specific amino acid sequence. Therefore, classical methods for determining whether a protein spans a membrane depend upon a *hydropathy profile*, which plots the hydrophobicity of the amino acids along the protein [6, 3, 7].

In this experiment, we measured the ability of an SVM classifier to recognize yeast membrane proteins. We use as a gold standard the annotations provided by the MIPS Comprehensive Yeast Genome Database (CYGD—`mips.gsf.de/proj/yeast`). The CYGD assigns subcellular locations to 2318 yeast proteins, of which 497 belong to various membrane protein classes.

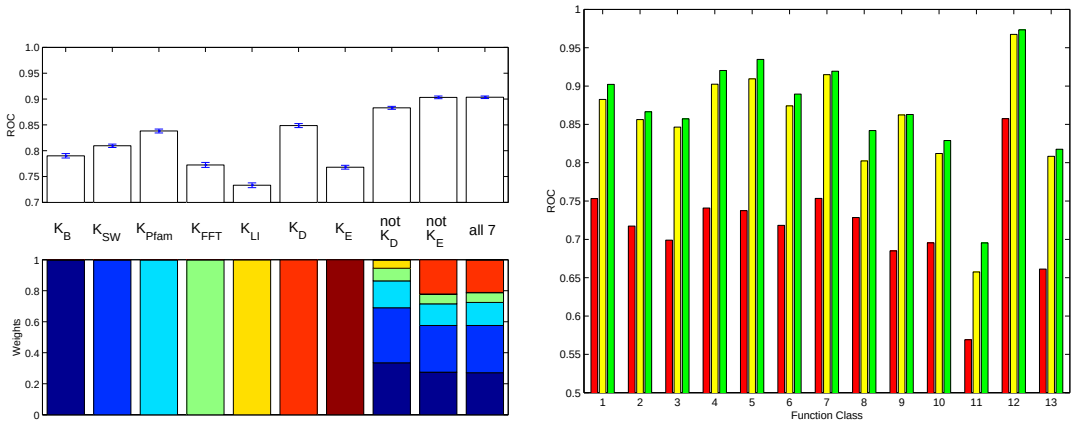
The primary input to the classification algorithm is a collection of seven kernel matrices. The first is a gene expression kernel K_E derived from 441 experiments from the Stanford Microarray Database (`genome-www.stanford.edu/microarray`). Three sequence-based kernels are included. Two are pairwise comparison kernels computed using BLAST and Smith-Waterman algorithms (K_{SW} and K_B, respectively). The third (K_{Pfam}) is similar, but is generated using E-values computed with respect to the Pfam database of protein domains (`pfam.wustl.edu`). Two protein-protein interaction kernels are used, a linear interaction kernel K_{LI} that computes scalar products on the rows of the matrix representation, and a diffusion kernel K_D computed on the interaction graph.

The final kernel K_{FFT} is specific to the membrane protein recognition task. It directly incorporates information about hydrophobicity patterns, which are known to be useful in identifying membrane proteins. The kernel uses hydropathy profiles generated from the Kyte-Doolittle index [9] and compares the frequency content of the hydropathy profiles of two proteins by using a fast Fourier transform algorithm and applying a Gaussian kernel function, $K(x_1, x_2) = \exp(-||x_1 - x_2||^2/2\sigma)$ with width $\sigma = 10$, to the corresponding vectors of FFT values. This kernel detects periodicities in the hydropathy profile, a feature that is relevant to the identification of membrane proteins.

The algorithm's performance is measured by splitting the data into a training and test set in a ratio of 80/20. We report the receiver operating characteristic (ROC) score, which is the area under a curve that plots true positive rate as a function of false positive rate for differing classification thresholds. The ROC score measures the overall quality of the ranking induced by the classifier, rather than the quality of a single point in that ranking. An ROC score of 0.5 corresponds to random guessing, and an ROC score of 1.0 implies that the algorithm succeeded in putting all of the positive examples before all of the negatives. Each experiment is repeated 30 times with different random splits in order to estimate the variance of the performance values.

The results are summarized in Figure 1(A). The plot illustrates that SDP/SVM learns significantly better from the heterogeneous data than from any single data type. The mean ROC score using all seven kernel matrices (0.9035 ± 0.0025) is significantly higher than the best ROC score using only one matrix (0.8487 ± 0.0039 using the diffusion kernel).

As expected, the sequence-based kernels yield good individual performance. Furthermore, when all seven matrices are used at once, the SDP assigns relatively large weights to the sequence-based kernels: $\mu_B = 1.90$, $\mu_{SW} = 2.13$, $\mu_{Pfam} = 1.04$, $\mu_{FFT} = 0.44$, $\mu_{LI} = 0.01$, $\mu_D = 1.47$ and $\mu_E = 0.01$ (for ease of interpretation, we scale the weights such that their sum is equal to the number m



(A) (B)

Figure 1: **Combining data sets yields better classification performance.** (A) Membrane protein recognition. The height of each bar is proportional to the ROC score of the SDP/SVM method using the given kernel. Error bars indicate standard error across 30 random train/test splits. The heights of the colored bars below each plot indicate the relative weight of the different kernel matrices in the optimal linear combination. (B) Protein functional classification. Red bars correspond to the MRF method of Deng *et al.*; yellow bars correspond to the SDP/SVM method using five kernels computed on binary data, and green bars correspond to the SDP/SVM using the enriched Pfam kernel and replacing the expression kernel with the SW kernel. The standard deviation across the 15 experiments is usually 0.01 or smaller, so most of the depicted differences are significant.

of kernel matrices). Thus, three of the four kernel matrices that receive weights larger than 1 are derived from the amino acid sequence. The SW kernel yields better results than the BLAST kernel, reflecting the fact that BLAST is a heuristic search procedure, whereas the SW algorithm finds the optimal local alignment. The results also show that the interaction-based diffusion kernel is more informative than the expression kernel. Not only has K_D an individual ROC score which is significantly higher than K_E, the SDP also assigns a weight of 1.47 to K_D, whereas K_E receives a weight of only 0.01. Accordingly, removing the diffusion kernel ("not K_D" in the plot) reduces the ROC score from 0.9035 to 0.8830, whereas removing the expression kernel ("not K_E") has almost no effect.

Finally, to test the robustness of our approach, a second experiment was performed in which a randomly generated kernel matrix K_{RND} was included (by computing inner products of random 100-element vectors, one for each protein). A control classifier trained using only K_{RND} yields an ROC score of 0.5, indicating K_{RND} is indeed uninformative here. More importantly, when a classifier is trained using all seven real kernels plus K_{RND}, SDP assigns K_{RND} a weight that is close to zero (0.12), and, as a consequence, the ROC score doesn't change significantly. This indicates that the method is robust to the presence of noisy, irrelevant data.

4.2 Yeast Protein Functional Classification

In a second experiment, we test our kernel-based approach by following the experimental paradigm of Deng *et al.*[5] The task is predicting functional classifications associated with yeast proteins, and we use as a gold standard the functional catalogue provided by the CYGD. The top-level categories in the functional hierarchy produce 13 classes. These 13 classes contain 3588 proteins; the remaining yeast proteins have uncertain function and are therefore not used in evaluating the classifier. Because a given protein can belong to several functional classes, we cast the prediction problem as 13 binary classification tasks, one for each functional class.

The primary input to the classification algorithm is a collection of kernel matrices representing different types of data. In order to compare the SDP/SVM approach to the Markov Random Field (MRF) method of Deng *et al.*, we perform two variants of the experiment: one in which the five kernels are restricted to contain precisely the same binary information as used by the MRF method, and a second experiment in which two of the kernels use richer representations and a sixth kernel is added.

The six kernels are as follows. K_{Pfam} summarizes the domain structure of the protein, as given by Pfam. Three kernels, K_{Gen}, K_{Phys}, and K_{TAP}, are diffusion kernels derived from observations of genetic, physical and TAP-induced physical interactions, as given in the CYGD. The fifth kernel uses 77 cell cycle gene expression measurements per gene. The final kernel is K_{SW}, as described above.

Each algorithm's performance is measured by performing 5-fold cross-validation three times. For a given split, we evaluate each classifier by reporting the ROC score on the test set. For each classification, we measure 15 ROC scores (three 5-fold splits), which allows us to estimate the variance of the score.

The experimental results are summarized in Figure 1(B). The figure shows that, for each of the 13 classifications, the ROC score of the SDP/SVM method is better than that of the MRF method. Overall, the mean ROC improves from 0.715 to 0.854. The improvement is consistent and statistically significant across all 13 classes. An additional improvement, though not as large, is gained by replacing

the expression and Pfam kernels with their enriched versions. The most improvement is offered by using the enriched Pfam kernel and replacing the expression kernel with the Smith-Waterman kernel. The resulting mean ROC is 0.870. Again, the improvement occurs in every class, although some class-specific differences are not statistically significant.

5 Discussion

We have described a general method for combining heterogeneous genome-wide data sets in the setting of kernel-based statistical learning algorithms, and we have applied it to the problems of predicting yeast membrane proteins and yeast protein function. The resulting SDP/SVM algorithm yields significant improvement relative to an SVM trained from any single data type and relative to a previously proposed graphical model approach for fusing heterogeneous genomic data. Moreover, the performance of the algorithm improves consistently in our experiments as additional genome-wide data sets are added to the kernel representation.

Kernel-based statistical learning methods have a number of general virtues as tools for biological data analysis. First, the kernel framework accommodates not only the vectorial and matrix data that are familiar in classical statistical analysis, but also more exotic data types such as strings and graphs. The ability to handle such data is clearly essential in the biological domain. Second, kernels provide significant opportunities for incorporating more specific biological knowledge, as we have seen with the FFT and Pfam kernel. Third, the growing suite of kernel-based data analysis algorithms require only that data be reduced to a kernel matrix; this creates opportunities for standardization. Finally, as shown here, the reduction of heterogeneous data types to the common format of kernel matrices allows the development of general tools for combining multiple data types. Kernel matrices are required only to respect the constraint of positive semidefiniteness, and thus the powerful technique of semidefinite programming can be exploited to derive general procedures for combining data of heterogeneous format and origin.

Acknowledgments

WSN is supported by a Sloan Foundation Research Fellowship and by National Science Foundation grants DBI-0078523 and ISI-0093302. MIJ and GL acknowledge support from ONR MURI N00014-00-1-0637 and NSF grant IIS-9988642.

References

[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. A basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[2] C. Berg, J. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions*. Springer, New York, 1984.

[3] S. D. Black and D. R. Mould. Development of hydrophobicity parameters to analyze proteins which bear post- or cotranslational modifications. *Anal. Biochem.*, 193:72–82, 1991.

[4] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT*, pages 144–152, Pittsburgh, PA, 1992. ACM Press.

[5] M. Deng, T. Chen, and F. Sun. An integrated probabilistic model for functional prediction of proteins. In *RECOMB*, pages 95–103, 2003.

[6] D. M. Engleman, T. A. Steitz, and A. Goldman. Identifying nonpolar transbilayer helices in amino acid sequences of membrane proteins. *Ann. Rev. Biophys. Biophys. Chem.*, 15:321–353, 1986.

[7] T. P. Hopp and K. R. Woods. Prediction of protein antigenic determinants from amino acid sequences. *Proc. Natl. Acad. Sci. USA*, 78:3824–3828, 1981.

[8] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In C. Sammut and A. Hoffmann, editors, *Proceedings of the International Conference on Machine Learning*. Morgan Kaufmann, 2002.

[9] J. Kyte and R. F. Doolittle. A simple method for displaying the hydropathic character of a protein. *Journal of Molecular Biology*, 157:105–132, 1982.

[10] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semi-definite programming. Technical Report CSD-02-1206, University of California, Berkeley, 2002.

[11] L. Liao and W. S. Noble. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. In *Proceedings of the Sixth Annual International Conference on Computational Molecular Biology*, pages 225–232, 2002.

[12] Y. Nesterov and A. Nemirovsky. *Interior-point polynomial methods in convex programming: Theory and applications*, volume 13 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, 1994.

[13] W. S. Noble. *Kernel methods in computational biology*, chapter Support vector machine applications in computational biology. MIT Press, 2004. In press.

[14] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[15] T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[16] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.