JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 9, Number 2, 2002

Pp. 401–411

Learning Gene Functional Classifications from Multiple Data Types

PAUL PAVLIDIS,[1] JASON WESTON,[2] JINSONG CAI,[3] and WILLIAM STAFFORD NOBLE[4]

ABSTRACT

In our attempts to understand cellular function at the molecular level, we must be able to synthesize information from disparate types of genomic data. We consider the problem of inferring gene functional classifications from a heterogeneous data set consisting of DNA microarray expression measurements and phylogenetic profiles from whole-genome sequence comparisons. We demonstrate the application of the support vector machine (SVM) learning algorithm to this functional inference task. Our results suggest the importance of exploiting prior information about the heterogeneity of the data. In particular, we propose an SVM kernel function that is explicitly heterogeneous. In addition, we describe feature scaling methods for further exploiting prior knowledge of heterogeneity by giving each data type different weights.

Key words: gene functional classification, phylogenetic profiles, microarray expression analysis.

1. INTRODUCTION

A PRIMARY GOAL IN BIOLOGY is to understand the molecular machinery of the cell. The sequencing projects currently underway provide one view of this machinery. A complementary view is provided by data from DNA microarray hybridization experiments. In this paper, we describe computational techniques for inferring gene function from these two distinct types of data. These techniques are a first step toward the longer-term goal of learning about gene function simultaneously from many different types of genomic data.

Clearly, the availability of complete genomic sequences of human and other species provides a tremendous opportunity for understanding the functions of biological macromolecules. In this work, we infer gene function from phylogenetic profiles that are derived from a comparison between a given gene and a collection of complete genomes. Each profile characterizes the evolutionary history of a given gene. There is evidence that two genes with similar phylogenetic profiles may have similar functions, the idea being that their similar pattern of inheritance across species is the result of a functional link (Pellegrini *et al.*, 1999).

[1]Columbia Genome Center, Columbia University, New York, NY 10027.
[2]BIOwulf Technologies LLC, 305 Broadway, 9th Floor, New York, NY 10007.
[3]Department of Medical Informatics, Columbia University, New York, NY, 10027.
[4]Department of Computer Science, Columbia University, New York, NY, 10027.

Gene function can also be inferred from DNA microarray expression data. By offering a snapshot of the messenger RNA expression levels of thousands of genes at once, microarrays allow biologists to formulate models of gene expression on a scale that was unimaginable several years ago. Initial analyses of this type of data focused on clustering algorithms, such as hierarchical clustering (Eisen *et al.*, 1998) and self-organizing maps (Tamayo *et al.*, 1999). These unsupervised algorithms attempt automatically to locate clusters of genes that share similar expression patterns and hence may share similarity in function. Subsequently, Brown *et al.* (2000) applied a collection of supervised learning techniques to a set of microarray expression data from yeast. They showed that an algorithm known as a support vector machine (SVM) (Boser *et al.*, 1992; Burges, 1998; Cristianini and Shawe-Taylor, 2000) provides excellent classification performance compared to a number of other methods, including Parzen windows, decision trees, and Fisher's linear discriminant (Brown *et al.*, 2000). SVMs are members of a larger class of algorithms known as kernel methods, which can be nonlinearly mapped to a higher-order feature space by replacing the inner product operation in the input space with a kernel function $K(\cdot, \cdot)$.

In this paper, we extend the methodology of Brown *et al.* to learn gene functional classifications from a heterogeneous data set consisting of microarray expression data and phylogenetic profiles. We test a variety of techniques for combining information from both data types, including a novel kernel function which is explicitly heterogeneous. We find that prior knowledge of heterogeneity can be exploited when selecting subsets of input features for use in classification. For most of the gene functional classifications that we investigated, one type of genomic data provides significantly better training data than the other type. Many feature selection algorithms are available for automatically selecting the most useful features to use in training a classifier. We demonstrate that, for these data, feature selection algorithms that select among data types (i.e., learn from phylogenetic profiles, from gene expression data, or from both) perform better than algorithms that directly select features from the combined data set. In this paper, in an extension of the methods described by Pavlidis *et al.* (2001b), we also show that one can obtain a further improvement in performance by weighting each data type according to its importance, rather than simply choosing among data types.

The idea of combining heterogeneous data sets to infer gene function is not new. Marcotte *et al.* (2000) describe an algorithm for functional annotation that uses expression vectors and phylogenetic profiles, as well as evolutionary evidence of domain fusion. However, the algorithm consists of predicting functional links between pairs of genes using each type of data separately and then cataloging the complete list of links. In contrast, the SVM method described here considers the various types of data at once, making a single prediction for each gene with respect to each functional category. Indeed, the performance of SVMs when data types are combined and a single hypothesis is formed is superior to combining two independent hypotheses, and we believe this will be true for a wide range of techniques.

2. METHODS

The experiments carried out here use two types of genomic data. The first data set derives from a collection of DNA microarray hybridization experiments (Eisen *et al.*, 1998). Each data point represents the logarithm of the ratio of expression levels of a particular gene under two different experimental conditions. The data consists of a set of 79-element gene expression vectors for 2,465 yeast genes. These genes were selected by Eisen *et al.* based on the availability of accurate functional annotations. The data were generated from spotted arrays using samples collected at various time points during the diauxic shift (DeRisi *et al.*, 1997), the mitotic cell division cycle (Spellman *et al.*, 1998), sporulation (Chu *et al.*, 1998), and temperature and reducing shocks, and are available on the Stanford web site (*www-genome.stanford.edu*).

In addition to the microarray expression data, each of the 2,465 yeast genes is characterized by a phylogenetic profile (Pellegrini *et al.*, 1999). In its simplest form, a phylogenetic profile is a bit string, in which the Boolean value of each bit indicates whether the gene of interest has a close homolog in the corresponding genome. The profiles employed in this paper contain, at each position, the negative logarithm of the lowest E-value reported by BLAST version 2.0 (Altschul *et al.*, 1997) in a search against a complete genome, with negative values (corresponding to E-values greater than 1) truncated to 0. Two genes in an organism can have similar phylogenetic profiles for one of two reasons. First, genes with a high level of sequence similarity will have, by definition, similar phylogenetic profiles. Second, for two genes which

lack sequence similarity, the similarity in phylogenetic profiles reflects a similar pattern of occurence of their homologs across species. This coupled inheritance may indicate a functional link between the genes, on the hypothesis that the genes are always present together or always both absent because they cannot function independently of one another. The profiles in this study are constructed using 24 complete genomes, collected from The Institute for Genomic Research website (*www.tigr.org/tdb*) and from the Sanger Centre website (*www.sanger.au.uk*). Prior to learning, the gene expression and phylogenetic profile vectors are adjusted to have a mean of 0 and a variance of 1.

Classification experiments are carried out using gene functional categories from the Munich Information Center for Protein Sequences Yeast Genome Database (MYGD) (*www.mips.biochem.mpg.de/proj/yeast*). The database contains several hundred functional classes, whose definitions come from biochemical and genetic studies of gene function. The experiments reported here use classes containing ten or more genes, and which are not substantially encompassed by any other class used, amounting to 108 classes. The complete data set and corresponding classifications are available at *www.cs.columbia.edu/compbio.*

For each class, a support vector machine is trained to discriminate between class members and non-members. A support vector machine is a supervised learning algorithm developed over the past decade by Vapnik and others (Boser *et al.*, 1992). In the form employed here, SVMs learn binary classifications; i.e., the SVM learns to answer the question "Does the given gene belong to functional class X?" where X is some category such as "ribosomal genes" or "sugar and carbohydrate transporters." Support vector machines classify points by locating them with respect to a hyperplane that separates class members from nonclass members in a high-dimensional feature space. The characteristics of the feature space are determined by a kernel function, which is selected a priori. Mercer's theorem (Mercer, 1909) shows that every positive semi-definite kernel function corresponds to the inner product operation in some higher-dimensional feature space. The current experiments employ a kernel function that has been shown to produce good classification performance for some MYGD classes using this gene expression data set (Brown *et al.*, 2000). The function is an inner product raised to the third power: $K(\vec{\mathbf{X}}, \vec{\mathbf{Y}}) = \left((\vec{\mathbf{X}} \cdot \vec{\mathbf{Y}}/\sqrt{\vec{\mathbf{X}} \cdot \vec{\mathbf{X}}}\sqrt{\vec{\mathbf{Y}} \cdot \vec{\mathbf{Y}}}) + 1\right)^3$. This kernel function takes into account pairwise and tertiary correlations among gene expression measurements. The normalization term in the denominator projects the data onto the unit sphere. We also test a radial basis kernel function, $K(X, Y) = exp(-||\vec{\mathbf{X}} - \vec{\mathbf{Y}}||^2/2\sigma^2)$. As in previous work, the SVM uses a soft margin that accounts for the disparity in the number of positive and negative examples for each class. For details about this adjustment, see Brown *et al.* (2000). A useful introduction to SVMs is available (Cristianini and Shawe-Taylor, 2000), as is the software used to perform these experiments (*www.cs.columbia.edu/compbio*).

The two types of data—gene expression and phylogenetic profiles—are combined in three different fashions, which we refer to as early, intermediate, and late integration. These three methods are summarized in Fig. 1. In early integration, the two types of vectors are concatenated to form a single set of length-103 vectors, which serve as input for the SVM. In intermediate integration, a heterogeneous kernel is used in which the kernel values for each type of data are precomputed separately, and the resulting values are added together. These summed kernel values are used in the training of the SVM. Thus, given the above kernel function $K(\cdot, \cdot)$, the heterogeneous kernel is $K(\vec{\mathbf{X}}_g, \vec{\mathbf{Y}}_g) + K(\vec{\mathbf{X}}_p, \vec{\mathbf{Y}}_p)$, where subscripts denote gene expression and phylogenetic profile data, respectively. Because the sum of any two positive semi-definite matrices is positive semi-definite, intermediate integration forms a valid kernel. Finally, in late integration, one SVM is trained from each data type, and the resulting discriminant values are added together to produce a final discriminant for each gene.

The heterogeneous kernel used in intermediate integration is an attempt to incorporate prior knowledge into the task at hand. The method creates local features that are polynomial relationships among inputs within a single type of data. These local features are then combined linearly to create global features. From these global features, a hyperplane is constructed. In constrast to the feature space produced by the early integration method, polynomial relationships among different types of inputs are now ignored. This restriction reflects our intuition that the correlations of inputs within one type of data are stronger than correlations between data types. In theoretical terms, removal of these correlations reduces overfitting as (unneeded) capacity is reduced. Indeed, this approach is similar in spirit to one already used in digit recognition problems in order to incorporate prior knowledge about spatial location (Vapnik, 1998). This incorporation was achieved by constructing sparse polynomials that sum across subkernels computed for many small patches within an image. In experiments on 400-pixel input spaces, the authors decreased the

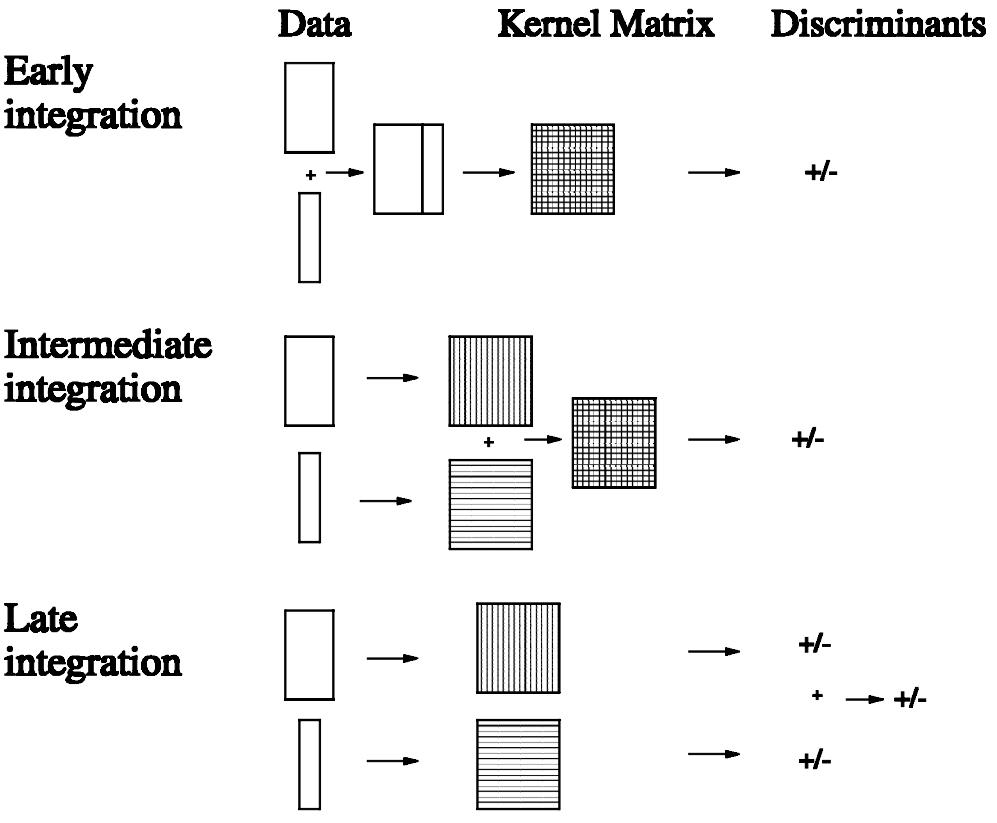

FIG. 1. Three methods for learning from heterogeneous data with a support vector machine. In early integration, the two types of data are concatenated to form a single set of input vectors. In intermediate integration, the kernel values are computed separately for each data set and then summed. In late integration, one SVM is trained on each data type, and the resulting discriminant values are summed.

number of polynomial features from 10^{23} to 10^{14} and reported a 68% reduction in test error (Burges, 1999).

Each classification experiment is performed using cross-validation. For a given class, the positively labeled and negatively labeled genes are split randomly into n groups for n-fold cross-validation. An SVM is trained on $n-1$ of the groups and is tested on the remaining group. This procedure is repeated n times, each time using a different group of genes as a test set. For most of the experiments, we use three-fold cross-validation. Leave-one-out cross-validation, which is used in some of the experiments, is simply n-fold cross-validation with n equal to the total number of training examples.

The performance of each SVM is measured by examining how well the classifier identifies the positive and negative examples in the test sets. To judge overall performance, we use the same measure as did Brown *et al.* (2000). This measure, the cost savings, allows one to assign a higher cost to false-negative classification errors compared to false positives. To see why this unequal cost weighting is important, consider two classifiers A and B trained to recognize the class of histones, which contains 15 genes. Say that, on a test set of 822 genes, classifier A correctly identifies all 15 histones but also includes 15 nonhistones in its list of positive genes. On the other hand, suppose that classifier B classifies everything in the test set as negative. Clearly, classifier A has learned to recognize something about the histones, whereas classifier B has learned nothing at all. However, using an equal weighting of false positive and false negatives, these two classifiers would yield the same cost (15). We assign a higher cost to false negatives in order to implement the intuition that failing to recognize one of the few positive examples is worse than inaccurately including one of the many negative examples in the test set. The cost savings is calculated as follows. We define the cost of using the method M as $C(M) = fp(M) + 2 \cdot fn(M)$, where $fp(M)$ is the number of false positives for method M, $fn(M)$ is the number of false negatives for method M, and n is the number of members in the class. The cost for each method is compared to the cost $C(N)$ for using the null

learning procedure, which classifies all test examples as negative. We define the normalized cost savings of using the learning procedure M as $S(M) = (C(N) - C(M))/2n$, where n is the total number of positive examples in the class. Thus, a perfect classifier has a normalized cost savings of 1, and the null classifier has a normalized cost savings of 0.

We also perform feature selection on the combined data using the Fisher criterion score (Bishop, 1995; Furey *et al.*, 2001). For a given feature j, we compute the mean and standard deviation of that feature across the positive examples (μ_j^+ and σ_j^+, respectively) and across the negative examples (μ_j^- and σ_j^-). The Fisher criterion score, $(\mu_j^+ - \mu_j^-)^2/((\sigma_j^+)^2 + (\sigma_j^-)^2)$, gives higher values to features whose means differ greatly between the two classes, relative to their variances. To perform feature selection, we determine the N features with the highest Fisher scores and form a vector $\vec{\mathbf{S}}$ of ones and zeros, indicating which features are selected and which are not. We then preprocess our training data with the operation $(\vec{\mathbf{X}} * \vec{\mathbf{S}})$, where the $*$ operator is element-wise multiplication of vectors. This procedure is equivalent to using the kernel $K(\vec{\mathbf{X}} * \vec{\mathbf{S}}, \vec{\mathbf{Y}} * \vec{\mathbf{S}})$ for support vector machine training.

In addition to feature selection, we consider feature scaling methods, in which features are weighted according to a real-valued vector $\vec{\mathbf{S}}$. We use a feature scaling method that incorporates prior knowledge about the heterogeneity of the data, weighting the two data types by the scalars s_1 and s_2. This technique reduces the preprocessing problem to finding just two real values. Selection of s_1 and s_2 can be done in many ways, e.g., choosing them by minimizing a theoretical bound or the error on a validation set. For computational reasons, we set s_1 and s_2 using the leave-one-out error of the nearest neighbor classifier, which performed well in the feature selection experiments that will later be summarized in Table 3. That is, we measure the cost savings from the leave-one-out procedure on the expression, phylogenetic profile, and combined data. Denoting these values as C_θ, C_p and C_c, we then choose $s_1 = \frac{1}{2} - \frac{1}{2}(C_c/(C_c + C_\theta)) + \frac{1}{2}(C_c/(C_c + C_p))$[1] and $s_2 = 1 - s_1$.

We also use the k-nearest neighbors algorithm (see, e.g., Duda and Hart [1973]) as a means of selecting a type of data from which to learn. The algorithm labels a test point as positive if more than $k/2$ of its closest (in Euclidean distance) neighbors from the training set are labeled positive; otherwise, the point is labeled negative. Taking $k = 1$ (i.e., 1-nearest neighbor) amounts to assigning to each point the label associated with its closest neighbor. Given a set of features, one can measure the quality of the set via the leave-one-out error of this algorithm.

3. RESULTS

Our initial experiments aimed at determining which classes among the 108 selected were learnable from either data type used alone. Based on the cost-savings measure, for each data type we selected the top 15 classes for further study. Three classes appear on both lists, yielding a total of 27 classes. The results are summarized in the first two columns of Table 1. Included in the table are the equivalents of all five classes used by Brown *et al.* (2000).[2] These experiments show that the SVM methodology generalizes well to phylogenetic profile data, and that this new type of data allows for the characterization of new functional classes. In other preliminary experiments, we found that a k-nearest neighbor classifier does much worse than the SVM, further motivating the use of the SVM approach (not shown; see *www.cs.columbia.edu/compbio/exp-phylo/* for data).

The second set of experiments tests the ability of the SVMs to learn from both types of data at once. The final five columns in Table 1 summarize these results, and Table 2 provides more details about the top five MYGD classes. Overall, integrating the data using a heterogeneous kernel function provides a normalized cost savings that is the best-performing or comparable to the best-performing method in 21 of

[1] In the case that some of the cost savings are negative, a constant is added to ensure positivity. The degenerative case of one of the denominators being equal to zero must also be avoided in this way.

[2] The MYGD functional catalog has been revised since the publication of Brown *et al.*, changing the composition of the classes substantially. For example, the "proteasome" class used in that paper has been subsumed into the "proteolysis" class. This is why the SVM performance on these classes differs from the performance reported in the earlier work.

Table 1. Summary of Learning Performance Results[a]

Class	*Expr*	*Phylo*	*Early*	*Intermd*	*Late*	*Early**	*Intermd**
amino acid transport	0.08 ± 0.02	**0.77 ± 0.10**	0.50 ± 0.04	**0.69 ± 0.07**	0.49 ± 0.08	**0.70 ± 0.07**	**0.72 ± 0.07**
ribosomal proteins	0.70 ± 0.02	0.08 ± 0.02	**0.75 ± 0.01**	0.72 ± 0.01	0.70 ± 0.01	**0.75 ± 0.00**	0.72 ± 0.01
sugar and carbohydrate transport	0.30 ± 0.07	0.67 ± 0.03	0.67 ± 0.05	**0.69 ± 0.01**	0.64 ± 0.02	**0.70 ± 0.02**	0.66 ± 0.01
glycolysis and gluconeogenesis	0.22 ± 0.04	0.43 ± 0.07	0.27 ± 0.02	**0.45 ± 0.04**	0.41 ± 0.04	0.38 ± 0.04	**0.49 ± 0.04**
mitochondrial org.	**0.41 ± 0.01**	0.15 ± 0.01	**0.42 ± 0.02**	**0.41 ± 0.02**	0.36 ± 0.01	**0.42 ± 0.01**	**0.42 ± 0.01**
tricarboxylic acid cycle	0.19 ± 0.16	0.17 ± 0.04	**0.31 ± 0.07**	**0.34 ± 0.09**	0.22 ± 0.09	**0.31 ± 0.09**	**0.34 ± 0.08**
deoxyribonucleotide metab.		**0.31 ± 0.12**	**0.24 ± 0.14**	**0.31 ± 0.12**	**0.31 ± 0.12**	0.17 ± 0.09	**0.26 ± 0.12**
org. of cytoplasm	0.35 ± 0.01	0.17 ± 0.01	**0.38 ± 0.00**	0.35 ± 0.02	0.35 ± 0.01	**0.37 ± 0.00**	0.36 ± 0.01
transport ATPases	0.14 ± 0.04	**0.36 ± 0.06**	0.22 ± 0.05	**0.36 ± 0.04**	0.23 ± 0.03	0.28 ± 0.05	**0.37 ± 0.08**
amino acid biosynthesis	0.16 ± 0.02	0.27 ± 0.02	0.29 ± 0.03	**0.34 ± 0.04**	0.27 ± 0.01	**0.34 ± 0.04**	**0.37 ± 0.03**
purine ribonucleotide metab.	0.17 ± 0.02	0.26 ± 0.05	0.19 ± 0.02	**0.33 ± 0.04**	0.17 ± 0.03	0.19 ± 0.04	**0.34 ± 0.05**
pyrimidine ribonucleotide metab.		0.32 ± 0.06	0.11 ± 0.04	0.26 ± 0.06	0.16 ± 0.03	0.20 ± 0.04	**0.42 ± 0.09**
cytoplasmic degradation	**0.33 ± 0.02**		**0.33 ± 0.02**	0.29 ± 0.03	0.18 ± 0.02	**0.33 ± 0.01**	0.29 ± 0.02
respiration	**0.30 ± 0.02**		**0.30 ± 0.03**	0.26 ± 0.03	0.17 ± 0.01	**0.28 ± 0.02**	0.26 ± 0.04
org. of chromosome structure	0.30 ± 0.01		0.30 ± 0.00	0.30 ± 0.00	0.12 ± 0.05	**0.31 ± 0.00**	0.29 ± 0.02
phosphate utililization	**0.22 ± 0.04**	0.08 ± 0.05	**0.26 ± 0.05**	**0.23 ± 0.07**	**0.22 ± 0.04**	**0.26 ± 0.05**	**0.25 ± 0.07**
org. of plasma membrane	0.06 ± 0.02	**0.25 ± 0.02**	0.23 ± 0.03	0.24 ± 0.01	**0.26 ± 0.01**	**0.26 ± 0.03**	**0.27 ± 0.01**
pentose phosphate pathway		0.20 ± 0.16		**0.22 ± 0.05**	0.17 ± 0.11	0.15 ± 0.05	**0.31 ± 0.08**
cellular import	0.03 ± 0.02	**0.24 ± 0.04**	0.17 ± 0.04	0.18 ± 0.02	0.20 ± 0.03	**0.21 ± 0.05**	**0.23 ± 0.04**
protein folding and stabilization		**0.24 ± 0.04**	0.19 ± 0.04	**0.24 ± 0.08**	0.16 ± 0.04	**0.24 ± 0.06**	**0.30 ± 0.07**
proteolysis	**0.23 ± 0.02**		**0.23 ± 0.01**	0.18 ± 0.05	0.15 ± 0.01	**0.23 ± 0.02**	0.18 ± 0.05
pheromone response	**0.23 ± 0.05**		0.15 ± 0.03	**0.18 ± 0.04**		**0.18 ± 0.07**	0.16 ± 0.09
nuclear org.	0.19 ± 0.01	0.07 ± 0.01	**0.23 ± 0.02**	**0.22 ± 0.02**	0.18 ± 0.01	**0.23 ± 0.02**	**0.22 ± 0.00**
drug-transporters		**0.22 ± 0.10**					
org. of endoplasmic reticulum	**0.20 ± 0.03**		**0.21 ± 0.03**	0.18 ± 0.03	0.12 ± 0.02	**0.19 ± 0.01**	0.17 ± 0.03
org. of cell wall	0.09 ± 0.07	**0.19 ± 0.07**	**0.13 ± 0.08**	**0.15 ± 0.09**	**0.21 ± 0.08**	**0.16 ± 0.09**	**0.16 ± 0.11**
anion transports		**0.21 ± 0.02**					
Best performing	7	9	12	15	4	19	16
Non-learnable	6	6	3	2	3	2	2
Mean cost value	0.18 ± 0.03	0.21 ± 0.04	0.26 ± 0.03	0.30 ± 0.04	0.24 ± 0.03	0.29 ± 0.04	0.32 ± 0.05

[a]Each row in the table contains the cost savings for one MYGD classification. Each cost savings is computed via three-fold cross-validation, with standard deviation calculated across five repetitions. The first two columns are from SVMs trained on a single type of data (gene expression or phylogenetic profiles). The remaining five columns are from SVMs trained using early, intermediate, late, early*, and intermediate* integration of the data, as described in the text. Values in bold face are the best-performing or are comparable to the best-performing of the five methods. A missing value indicates that the cost savings is not significantly greater than zero. The last three rows are summary statistics, giving the average values for each method and total number of bold-face and missing values in each column.

TABLE 2. ERROR RATES FOR SELECTED MYGD CLASSES[a]

Class	*Size*	*FP*	*FN*
amino acid transporters	22	2.0 ± 0.4	5.6 ± 0.2
ribosomal proteins	173	26.6 ± 1.2	34.2 ± 1.1
sugar and carbohydrate transporters	32	2.4 ± 0.7	9.0 ± 0.0
deoxyribonucleotide metabolism	9	0.2 ± 0.2	4.6 ± 0.7
mitochondrial organization	296	84.8 ± 1.8	128.4 ± 1.7

[a]The table lists error rates for the five most learnable MYGD classes. Each row contains the name and size of the class, and the average numbers of false positives and false negatives for that class from an SVM using intermediate integration.

the 27 classes, where "comparable" means within one standard deviation of the best-performing method. This is more classes than any of the other four methods. Furthermore, the average cost savings across all classes is higher for this method than any other ($p < 0.05$, paired Student's t-test). Similarly, the intermediate integration scheme fails to learn to classify only two classes, which is fewer classes than any of the other methods. Using a radial basis function kernel instead of a third-degree polynomial, we obtained similar results (not shown; see *www.cs.columbia.edu/compbio/exp-phylo/* for data).

Learning from both data types is not always a good idea. For four classes, all three of the combined methods lead to decreased classification performance relative to an SVM trained on a single type of data. In every case, the decrease occurs when one data type provides much more information than the other, indicating that the inferior data type contributes noise that disrupts learning. This observation suggests that a feature selection algorithm that effectively eliminates noisy features should allow an SVM to learn these classes more accurately.

However, our experiments demonstrate that a naive feature selection algorithm, which does not take into account the heterogeneity of the data, does not typically yield improved classification performance. Figure 2 shows the results of using the Fisher criterion score to select features from the combined data.

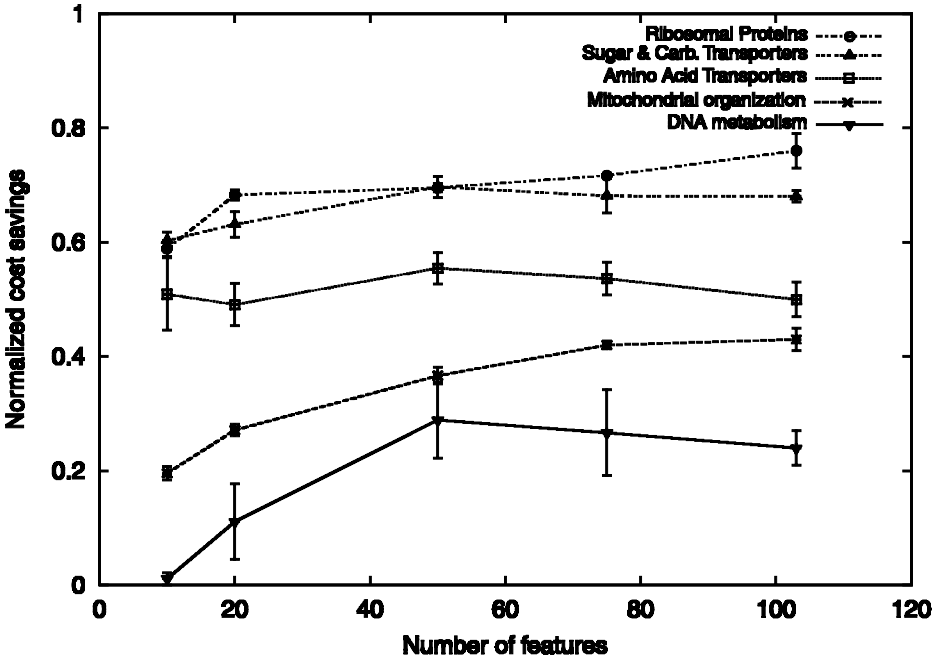

FIG. 2. Effect of feature selection on learning performance. Each series shows the performance of an SVM trained on the combined data set using varying numbers of features, selected according to the Fisher criterion score. These examples are representative of results obtained for all 27 classes tested.

Table 3. Feature Selection for Optimal Data Set Choice[a]

	Express	*Phylo*	*Early*	*Fisher sel*	*1-NN sel*	*5-CV sel*
wins in all 3 trials	8	15	17	19	20	20
wins in 2 or more trials	8	15	17	22	25	26

[a]The table lists the number of classes for which various data set selection algorithms choose the best or within one standard deviation of the best performing data set. The results are computed over three separate trials, so the values are given for the case when the best choice is made in all three trials or in more than two trials. The algorithms (columns from left to right) are: always choosing the expression, phylogenetic or combined data, and selecting the dataset via the Fisher criterion score, leave-one-out error of the 1-nearest neighbor algorithm and five-fold cross-validation of SVMs.

By treating each feature independently, this simple feature selection algorithm does not take into account possible correlations between features, but the algorithm has the advantages of simplicity and efficiency. For the most part, classification performance declines as features are removed. For several MYGD classes, when combining the two types of data leads to a substantial decline in performance, feature selection yields a small improvement. However, in none of these cases does feature selection yield a level of performance comparable to that obtained using a single data type.

A more sophisticated feature selection method, which does take into account correlations and nonlinearities, yields similar results. The method uses the SVM solution to measure the quality of the features and removes features that appear to contribute least—a so-called filter method (see, e.g., Weston *et al.* [2001]). We do not give details of the method here because, although the results are marginally superior to the Fisher criterion results, the same problem arises: the SVM feature selection method does not achieve performance equal to the best-performing single data type in cases where the combined data set performs poorly. Even worse, for these cases, hand-picking the best-performing single data set and trying various ways of adding features to it still leads to a consistent deterioration in performance. Apparently, when one data set performs quite poorly compared to the other (e.g., amino acid transporters, glycolysis and glucogenesis, sugar and carbohydrate transporters, etc.), no useful information can be gleaned from it.

The problem that the combined data sometimes performs poorly (or, in general, one dataset/kernel combination performs better than others) suggests that it would be useful to determine that such an outcome was likely before attempting to train. In other words, one would like to be able to make the optimal choice between the options of using one data type, or the combination, before training.

Because this task is a special case of feature selection (selecting the best set of features given n distinct sets), traditional quality measures for feature selection can still be used. For example, the Fisher criterion score can be used to select the best data set by choosing the largest of the mean Fisher criterion scores.[3] One can also attempt to estimate the SVM's performance directly, via generalization bounds such as the VC bounds (Vapnik, 1998) or the span bound (Chapelle and Vapnik, 2000), or via cross-validation. We found that only the cross-validation was useful: the VC bound and span bound are too loose in this case. Finally, we also measure the leave-one-out error of the nearest neighbor algorithm, which can be considered as a compromise between the Fisher criterion score and cross-validation in terms of computation speed versus accuracy.

We tested this idea by selecting among three data sets: the gene expression data, phylogenetic profiles, or concatenated data set. This operation can be characterized by the preprocessing $s_1 K(\vec{\mathbf{X}}_e, \vec{\mathbf{Y}}_e) + s_2 K(\vec{\mathbf{X}}_p, \vec{\mathbf{Y}}_p)$ (in the intermediate integration case), where the free parameters are two binary variables s_1 and s_2. For the 27 classes, we counted how many times in three trials each method chose the best-performing data set. These results, shown in Table 3, indicate that choosing the correct data set ahead of time can give improved results. Both cross-validation and nearest neighbors are superior to the Fisher criterion score, since even when these methods do not choose the outright best data set they choose one close to the best (as hinted by the "wins in two or more trials" results). However, while cross-validation gains this improvement at the expense of high computational cost, nearest neighbors provides a cheap compromise. Bearing in mind

[3]The mean Fisher score of the combined data set is always intermediate between the mean Fisher scores from the two subsets. Hence, we choose the combined data set if its score falls within 10% of the highest alternative.

that nearest neighbors can work in a feature space via kernels (Cristianini and Shawe-Taylor, 2000), these preliminary results suggest that one can estimate the performance of any kernel (including the intermediate integration method) with fairly low computational cost.

Motivated by the improved performance shown by the above data set selection technique, we also tested a data set scaling technique. As described in the methods section, an alternative to feature selection is feature scaling. We performed data type scaling on both the early integration and intermediate integration methods. In the early integration case, this technique amounts to employing the kernel $K([s_1\vec{\mathbf{X}}_e\ s_2\vec{\mathbf{X}}_p], [s_1\vec{\mathbf{Y}}_e\ s_2\vec{\mathbf{Y}}_p])$ (where square brackets indicate concatenation of vectors), and in the intermediate integration case, the kernel is $s_1K(\vec{\mathbf{X}}_e a, \vec{\mathbf{Y}}_e) + s_2K(\vec{\mathbf{X}}_p, \vec{\mathbf{Y}}_p)$. We refer to these algorithms as Early* and Intermediate*. The results of using these algorithms on the 27 classes are given in the last columns of Table 1. Applied to either the early or intermediate integration method, this weighting procedure yields improved results based on both the mean cost savings and the number of top-scoring classes.

4. DISCUSSION

As the quantity and variety of genomic data increases, molecular biology shifts from a hypothesis-driven model to a data-driven one. Whereas previously a single laboratory could collect data and test hypotheses regarding a single system or pathway, this new paradigm requires combining genome-wide experimental results, typically gathered and shared across multiple laboratories. For example, constructing a single, n-element phylogenetic profile requires the availability of n complete genomic sequences, which clearly could not yet be generated by a single laboratory. The data-driven model requires sophisticated computational techniques that handle very large, heterogeneous data sets.

The support vector machine learning algorithm is such a technique. SVMs scale well and have been used successfully with large training sets in the domains of text categorization and image recognition (Cristianini and Shawe-Taylor, 2000). Furthermore, in this paper, we demonstrate that SVMs can learn from heterogeneous data sets. With an appropriate kernel function, the SVM learns from a combination of two different types of feature vectors. In most cases, the resulting trained SVM provides as good or better gene functional classification performance than an SVM trained on either data set alone.

For these data and these classifications, the heterogeneous kernel we introduce here (the intermediate integration method) performs somewhat better than the other techniques we investigated. We hypothesize that the improved performance results from the kernel's ability to exploit our prior knowledge that correlations within one type of data are stronger than correlations between different types. However, our results with data type selection and scaling show that it is even more important to determine the importance of each type of data for a given class, because even the heterogeneous kernel does not provide the best performance across all classes. By weighting each data type according to its importance for classification, we make better use of this prior knowledge.

Our results show that the supervised learning methodology proposed by Brown *et al.* (2000) can be extended in a straightforward fashion to some other classes in the MYGD. We have also shown, however, that the majority of the MYGD classifications are not learnable from either gene expression data or phylogenetic profiles. We do not believe that the failure to learn many of these classes is a failure of the SVM method. Rather, for many functional classes, the data are simply not informative. The expression data is only informative if the genes in the class are coordinately regulated at the level of transcription under the conditions tested. Similarly, phylogenetic profiles are limited in resolution in part because relatively few genomes are available. In particular, among the genomes from which we derived the phylogenetic profiles, all but one are bacterial. Thus it is difficult to generate useful phylogenetic profiles for genes that are specific to eucaryotes. Because the availability of expression, sequence, and other kinds of data are increasing steadily, we expect that the tools we are developing will continue to improve in power.

In our experiments, the primary utility of the phylogenetic profiles appears to lie in their ability to summarize sequence similarity information rather than the inheritance patterns of genes during various speciation events. Analysis of the classes that are most easily learnable from phylogenetic profiles alone shows that these are also the classes that share considerable sequence similarity among their members, for example, the various transporter classes (data not shown). In a previously published report on the use of phylogenetic profiles in yeast (Marcotte *et al.*, 1999), this effect was eliminated by merging groups

of similar genes and by making links between pairs of genes, rather than requiring that the phylogenetic profile similarity extend throughout an entire functional class, as we have. In our experiments, removing or combining the data for genes with sequence similarity would have had the undesirable effect of forcing the combining of the corresponding expression data, and there is no reason to think that genes with sequence similarity would generally be coordinately regulated at the expression level. However, there are obviously benefits to considering sequence similarity in a gene classification task, and in the future we will consider other techniques for summarizing sequence similarities in a fixed-length vector.

In most of the experiments reported here, we used the third-degree polynomial kernel function. This consistency allows direct comparisons across different feature combination and feature selection algorithms. We selected this particular kernel because of its straightforward interpretation (accounting for all primary features and up to tertiary correlations in the data) and because previous work showed that this kernel performs well for this gene expression data set (Brown *et al.*, 2000). Another kernel that performs well is the radial basis kernel, and our experiments with this kernel indicate that similar improvements can be gained by incorporating knowledge of heterogeneity. We note that the performance of the various algorithms would no doubt be improved by empirical kernel optimization. There is no reason to suppose, however, that a particular method would benefit more than others from such optimization.

The experiments reported here suggest several avenues for future research. One obvious research direction involves including additional types of data. Having shown that two types of data can be fruitfully combined, we plan to extend the techniques described here to feature vectors derived from, for example, the upstream promoter regions of genes (Pavlidis *et al.*, 2001a). We also plan to experiment with the Fisher kernel method, in which each type of data is compared to a probabilistic model of the domain (Jaakkola and Haussler, 1998; Jaakkola *et al.*, 1999). By converting the heterogeneous features to probability gradients, we hope to make the various types of data more directly comparable.

Support vector machines are part of a larger class of algorithms known as kernel methods, which have recently been gaining in popularity (Schölkopf *et al.*, 1999). A kernel method is any algorithm that employs a kernel function to implicitly operate in a higher-dimensional space. In addition to SVM classifiers, kernel methods have been developed for regression (Schölkopf *et al.*, 1996), discriminant analysis (Mika *et al.*, 1999), and principal components analysis (Schölkopf *et al.*, 1997). More members of this promising class of algorithms should be applied to problems in computational biology.

ACKNOWLEDGMENTS

The authors wish to thank Ilya Muchnik and Vladimir Vapnik for helpful discussions. W.N.G. is funded by an Award in Bioinformatics from the PhRMA Foundation and by National Science Foundation grants DBI-0078523 and IIS-0093302.

REFERENCES

Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D.J. 1997. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucl. Acids Res.* 25, 3389–3402.

Bishop, C. 1995. *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, UK.

Boser, B.E., Guyon, I.M., and Vapnik, V. 1992. A training algorithm for optimal margin classifiers. *Proc. 5th Annual ACM Workshop on Computational Learning Theory*, 144–152.

Brown, M.P.S., Grundy, W.N., Lin, D., Cristianini, N., Sugnet, C., Furey, T.M., Ares, J., and Haussler, D. 2000. Knowledge-based analysis of microarray gene expression data using support vector machines. *Proc. Natl. Acad. Sci. USA* 97(1), 262–267.

Burges, C.J.C. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery* 2(2), 121–167.

Burges, C.J.C. 1999. Geometry and invariance in kernel based methods. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, eds. *Advances in Kernel Methods—Support Vector Learning*, 89–116, MIT Press, Cambridge, MA.

Chapelle, O., and Vapnik, V. 2000. Model selection for support vector machines. In S.A. Solla, T.K. Leen, and K.-R. Müller, eds. *Advances in Neural Information Processing Systems 12*, MIT Press.

Chu, S., DeRisi, J., Eisen, M., Mulholland, J., Botstein, D., Brown, P., and Herskowitz, I. 1998. The transcriptional program of sporulation in budding yeast. *Science* 282, 699–705.

Cristianini, N., and Shawe-Taylor, J. 2000. *An Introduction to Support Vector Machines*, Cambridge University Press, Cambridge, UK.

DeRisi, J., Iyer, V., and Brown, P. 1997. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science* 278, 680–686.

Duda, R.O., and Hart, P.E. 1973. *Pattern Classification and Scene Analysis*, Wiley, New York.

Eisen, M., Spellman, P., Brown, P., and Botstein, D. 1998. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA* 95, 14863–14868.

Furey, T.S., Cristianini, N., Duffy, N., Bednarski, D.W., Schummer, M., and Haussler, D. 2001. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 906–914.

Jaakkola, T., Diekhans, M., and Haussler, D. 1999. Using the Fisher kernel method to detect remote protein homologies. *Proc. 7th Int. Conf. Intelligent Systems for Molecular Biology*, 149–158.

Jaakkola, T., and Haussler, D. 1998. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems 11*, Morgan Kauffmann, San Mateo, CA.

Marcotte, E.M., Pellegrini, M., Thompson, M.J., Yeates, T.O., and Eisenberg, D. 1999. A combined algorithm for genome-wide prediction of protein function. *Nature* 402(6757), 83–86.

Mercer, J. 1909. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, A209, 415–446.

Mika, S., Rätsch, G., Weston, J., Schölkopf, B., and Müller, K.-R. 1999. Fisher discriminant analysis with kernels. *Proc. IEEE Neural Networks for Signal Processing Workshop 1999.*

Pavlidis, P., Furey, T.S., Liberto, M., Haussler, D., and Grundy, W.N. 2001a. Promoter region-based classification of genes. *Proc. Pacific Symposium on Biocomputing*, 151–163.

Pavlidis, P., Weston, J., Cai, J., and Grundy, W.N. 2001b. Gene functional classification from heterogeneous data. *RECOMB'01*, 242–248.

Pellegrini, M., Marcotte, E.M., Thompson, M.J., Eisenberg, D., and Yeates, T.O. 1999. Assigning protein functions by comparative genome analysis: Protein phylogenetic profiles. *Proc. Natl. Acad. Sci. USA* 96(8), 4285–4288.

Schölkopf, B., Burges, C.J.C., and Smola, A.J., eds. 1999. *Advances in Kernel Methods: Support Vector Learning*, MIT Press, Cambridge, MA.

Schölkopf, B., Smola, A., and Müller, K.-R. 1996. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation* 10(5), 1299–1319.

Schölkopf, B., Smola, A., and Müller, K.-R. 1997. Kernel principal component analysis. In *Proceedings ICANN97*, Springer Lecture Notes in Computer Science, p. 583.

Spellman, P.T., Sherlock, G., Zhang, M.Q., Iyer, V.R., Anders, K., Eisen, M.B., Brown, P.O., Botstein, D., and Futcher, B. 1998. Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Mol. Biol. Cell* 9, 3273–3297.

Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E., and Golub, T. 1999. Interpreting patterns of gene expression with self-organizing maps. *Proc. Natl. Acad. Sci. USA* 96, 2907–2912.

Vapnik, V.N. 1998. *Statistical Learning Theory. Adaptive and Learning Systems for Signal Processing, Communications, and Control*, Wiley, New York.

Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., and Vapnik, V. 2001. Feature selection for SVMs. In S.A. Solla, T.K. Leen, and K.-R. Müller, eds. *Advances in Neural Information Processing Systems 13*, MIT Press, Cambridge, MA.

Address correspondence to:
William Stafford Noble
Dept. of Computer Science
Columbia University
450 Computer Science Bldg.
Mail Code 0401
1214 Amsterdam Ave.
New York, NY 10027

E-mail: noble@cs.columbia.edu