Support Vector Machine Classification of Microarray Gene Expression Data

UCSC-CRL-99-09

Michael P. S. Brown[†]
William Noble Grundy[†] [*]
David Lin[†]
Nello Cristianini[§] [†]
Charles Sugnet[‡]
Manuel Ares, Jr.[‡]
David Haussler[†]

[†]Department of Computer Science
University of California, Santa Cruz
Santa Cruz, CA 95065
{mpbrown,bgrundy,dave,haussler}@cse.ucsc.edu

[‡]Center for Molecular Biology of RNA
Department of Biology
University of California, Santa Cruz
Santa Cruz, CA 95065

[§]Department of Engineering Mathematics
University of Bristol
Bristol, UK

June 12, 1999

[*]Corresponding author: Department of Computer Science, Columbia University, 450 Computer Science Building, Mail Code 0401, 1214 Amsterdam Avenue, New York, NY 10027
[†]Work done while visiting UCSC.

Abstract

We introduce a new method of functionally classifying genes using gene expression data from DNA microarray hybridization experiments. The method is based on the theory of support vector machines (SVMs). We describe SVMs that use different similarity metrics including a simple dot product of gene expression vectors, polynomial versions of the dot product, and a radial basis function. Compared to the other SVM similarity metrics, the radial basis function SVM appears to provide superior performance in identifying sets of genes with a common function using expression data. In addition, SVM performance is compared to four standard machine learning algorithms. SVMs have many features that make them attractive for gene expression analysis, including their flexibility in choosing a similarity function, sparseness of solution when dealing with large data sets, the ability to handle large feature spaces, and the ability to identify outliers.

Keywords: Gene Microarrays, Gene Expression, Support Vector Machines, Pattern Classification, Functional Gene Annotation
Running head: SVM Classification of Gene Expression Data

1 Introduction

The advent of DNA microarray technology provides biologists with the ability to measure the expression levels of thousands of genes in a single experiment. Initial experiments [Eisen et al., 1998] suggest that genes of similar function yield similar expression patterns in microarray hybridization experiments. As data from such experiments accumulates, it will be essential to have accurate means for extracting its biological significance and for assigning functions to genes.

Currently, most approaches to the computational analysis of gene expression data attempt to learn functionally significant classifications of genes in an *unsupervised* fashion. A learning method is considered unsupervised if it learns in the absence of a teacher signal that provides prior knowledge of the correct answer. Existing gene expression analysis methods begin with a definition of similarity (or a measure of distance) between expression patterns, but with no prior knowledge of the true functional classes of the genes. Genes are then grouped using a clustering algorithm such as hierarchical clustering [Eisen et al., 1998, Spellman et al., 1998b] or self-organizing maps [Tamayo et al., 1999].

Support vector machines (SVMs) [Vapnik, 1998, Burges, 1998, Scholkopf et al., 1999] and other supervised learning techniques adopt the opposite approach. SVMs have been successfully applied to a wide range of pattern recognition problems, including handwriting recognition, object recognition, speaker identification, face detection and text categorization [Burges, 1998]. SVMs are attractive because they boast an extremely well developed theory. A support vector machine finds an optimal separating hyperplane between members and non-members of a given class in an abstract space. SVMs, as applied to gene expression data, begin with a collection of known classifications of genes. These collections, such as genes coding for ribosomal proteins or genes coding for components of the proteasome, contain genes known to encode proteins that function together and hence exhibit similar expression profiles. One could build a classifier capable of discriminating between members and non-members of a given class, such that, given expression data for a particular gene, one would be able to answer such questions as, "Does this gene code for a ribosomal protein?" Such a classifier would be useful in recognizing new members of the class among genes of unknown function. Furthermore, the classifier could be applied to the original set of training data to identify outliers that may have been previously unrecognized. Whereas unsupervised methods determine how a set of genes clusters into functional groups, SVMs determine what expression characteristics of a given gene make it a part of a given functional group. Because the question asked by supervised methods is much more focused than the corresponding question asked by unsupervised methods, supervised methods can use complex models that exploit the specific characteristics of the given functional group.

We describe the first use of SVMs to classify genes based on gene expression. We analyze expression data from 2467 genes from the budding yeast *S. cerevisiae* measured in 79 different DNA microarray hybridization experiments [Eisen et al., 1998]. From these data, we learn five functional classifications from the MIPS Yeast Genome Database (MYGD) [MYGD, 1999]. In addition to SVM classification, we subject these data to analyses by four competing machine learning techniques, including Fisher's linear discriminant [Duda and Hart, 1973], Parzen windows [Bishop, 1995], and two decision tree learners [Quinlan, 1997, Wu et al., 1999]. The SVM method significantly outperforms all other methods investigated here. Furthermore, investigation of genes where the SVM classification differs from the MIPS classification reveals many interesting borderline cases and some plausible mis-annotations in MIPS.

2 DNA microarray data

Each data point produced by a DNA microarray hybridization experiment represents the ratio of expression levels of a particular gene under two different experimental conditions. An experiment starts with microarray construction, in which several thousand DNA samples are fixed to a glass slide, each at a known position in the array. Each sequence corresponds to a single gene within the organism under investigation. Messenger RNA samples are then collected from a population of cells subjected to various experimental conditions. These samples are converted to cDNA via reverse transcription and are labeled with one of two different fluorescent dyes in the process. A single experiment consists of hybridizing the microarray with two differently labeled cDNA samples collected at different times. Generally, one of the samples is from the reference or background state of the cell, while the other sample represents a special condition set up by the experimenter, for example, heat shock. The level of expression of a particular gene is roughly proportional to the amount of cDNA that hybridizes with the DNA affixed to the slide. By measuring the ratio of each of the two dyes present at the position of each DNA sequence on the slide using laser scanning technology, the relative levels of gene expression for any pair of conditions can be measured [Lashkari et al., 1997, DeRisi et al., 1997]. The result, from an experiment with n DNA samples on a single chip, is a series of n expression-level ratios. Typically, the numerator of each ratio is the expression level of the gene in the condition of interest to the experimenter, while the denominator is the expression level of the gene in the reference state of the cell.

The data from a series of m such experiments may be represented as a gene expression matrix, in which each of the n rows consists of an m-element expression vector for a single gene. In our experiments the number of experiments m is 79 and the number of genes n is 2467. Following Eisen *et al.*, we do not work directly with the ratio as discussed above but rather with its logarithm[Eisen et al., 1998]. We define X_i to be the logarithm of the ratio of gene X's expression level in experiment i to X's expression level in the reference state. This log ratio is positive if the gene is induced (turned up) with respect to the background and negative if it is repressed (turned down).

The goal of our SVM classifier is to determine accurately the functional class of a given gene based only upon its expression vector $\mathbf{X}$. Visual inspection of the raw data indicates that such classification should be possible. Figure 1 shows the expression vectors for 121 yeast genes that participate in the cytoplasmic ribosome. The similarities among the expression vectors is clear.

It should be noted that although the mRNA expression vectors in Figure 1 are plotted left-to-right as functions, this is only as a visual convenience. The total mRNA expression data for a gene is not a single times series, but rather a concatenation of different, independent mRNA expression measurements, some of which happen to be clustered in time. Our focus here is on how to analyze large mRNA data sets such as this, which combine information from many unrelated microarray experiments. For this reason we do not explore Fourier transform or other times series oriented feature extraction methods here, e.g. as in [Spellman et al., 1998b], although further preprocessing of the mRNA measurements to remove bad data and reduce the noise in the short time series included in them may have been helpful, and will be considered in future work.

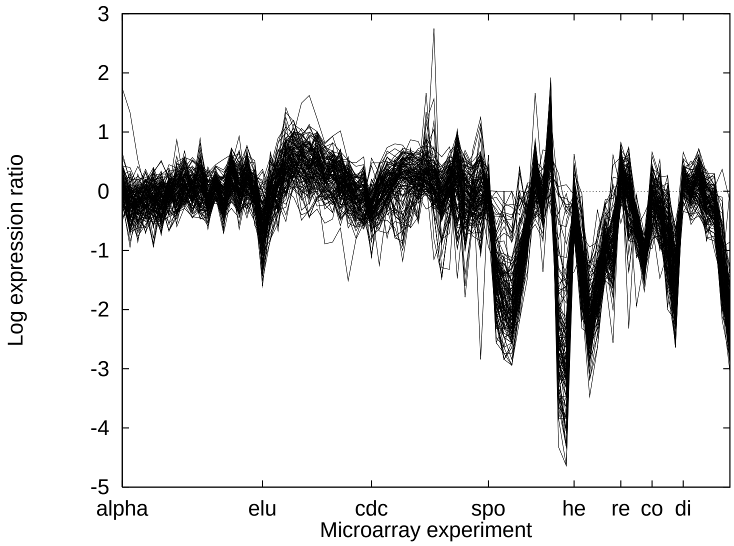
3 Support vector machines

Each vector in the gene expression matrix may be thought of as a point in an m-dimensional space. A simple way to build a binary classifier is to construct a hyperplane separating class members from non-members in this space. This is the approach taken by perceptrons, also known as single-layer neural networks.

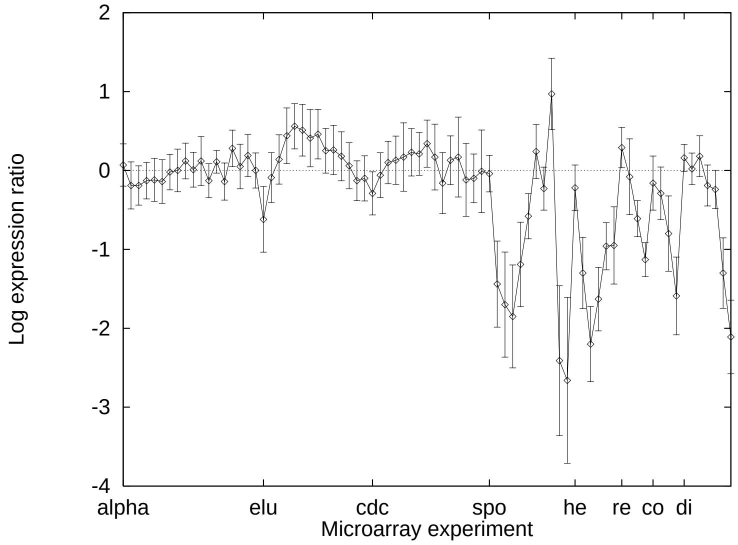
Unfortunately, most real-world problems involve non-separable data for which there does not exist a hyperplane that successfully separates the class members from non-class members in the training set. One solution to the inseparability problem is to map the data into a higher-dimensional space and define a separating hyperplane there. This higher-dimensional space is called the *feature space*, as opposed to the *input space* occupied by the training examples. With an appropriately chosen feature space of sufficient dimensionality, any consistent training set can be made separable.

However, translating the training set into a higher-dimensional space incurs both computational and learning-theoretic costs. Representing the feature vectors corresponding to the training set can be extremely expensive in terms of memory and time. Furthermore, artificially separating the data in this way exposes the learning system to the risk of finding trivial solutions that overfit the data.

Support vector machines elegantly sidestep both difficulties [Vapnik, 1998]. SVMs avoid overfitting by choosing a specific hyperplane among the many that can separate the data in the feature space. SVMs find the *maximum margin hyperplane*, the hyperplane that maximixes the minimum distance from the hyperplane to the closest training point (see Figure 2). The maximum margin hyperplane can be represented as a linear combination of training points. Consequently, the decision function for classifying points with respect to the hyperplane only involves dot products between points. Furthermore, the algorithm that finds a separating hyperplane in the feature space can be stated entirely in terms of vectors in the input space and dot products in the feature space. Thus, a support vector machine can locate a separating hyperplane in the feature space and classify points in that space without ever representing the space explicitly, simply by defining a function, called a *kernel function*, that plays the role of the dot product in the feature space. This technique avoids the computational burden of explicitly representing the feature vectors.



(a)



(b)

Figure 1: **Expression profiles of the cytoplasmic ribosomal proteins.** Figure (a) shows the expression profiles from the data in [Eisen et al., 1998] of 121 cytoplasmic ribosomal proteins, as classified by MYGD [MYGD, 1999]. The logarithm of the expression ratio is plotted as a function of DNA microarray experiment. Ticks along the X-axis represent the beginnings of experimental series. They are, from left to right, cell division cycle after synchronization with α factor arrest (alpha), cell division cycle after synchronization by centrifugal elutriation (elu), cell division cycle measured using a temperature sensitive *cdc15* mutant (cdc), sporulation (spo), heat shock (he), reducing shock (re), cold shock (co), and diauxic shift (di). Sporulation is the generation of a yeast spore by meiosis. Diauxic shift is the shift from anaerobic (fermentation) to aerobic (respiration) metabolism. The medium starts rich in glucose, and yeast cells ferment, producing ethanol. When the glucose is used up, they switch to ethanol as a source for carbon. Heat, cold, and reducing shock are various ways to stress the yeast cell. Figure (b) shows the average, plus or minus one standard deviation, of the data in Figure (a).

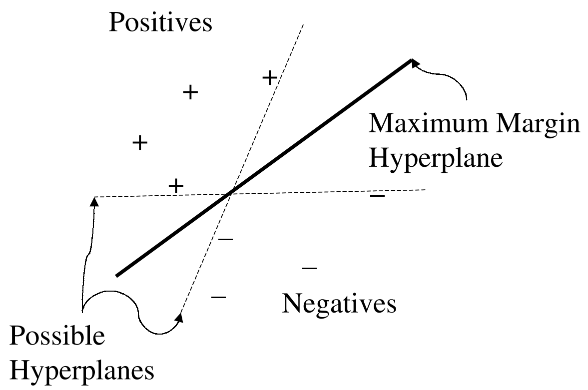


Figure 2: **Maximum margin hyperplane.** The figure shows four positive and four negative examples in a two-dimensional input space. Three separating hyperplanes are shown, including the maximum margin hyperplane.

The selection of an appropriate kernel function is important, since the kernel function defines the feature space in which the training set examples will be classified. As long as the kernel function is legitimate, an SVM will operate correctly even if the designer does not know exactly what features of the training data are being used in the kernel-induced feature space. The definition of a legitimate kernel function is given by Mercer's theorem [Vapnik, 1998]: the function must be continuous and positive definite. Human experts often find it easier to specify a kernel function than to specify explicitly the training set features that should be used by the classifier. The kernel expresses prior knowledge about the phenomenon being modeled, encoded as a similarity measure between two vectors.

In addition to counteracting overfitting, the SVM's use of the maximum margin hyperplane leads to a straightforward learning algorithm that can be reduced to a convex optimization problem. In order to train the system, the SVM must find the unique minimum of a convex function. Unlike the backpropagation learning algorithm for artificial neural networks, a given SVM will always deterministically converge to the same solution for a given data set, regardless of the initial conditions. For training sets containing less than approximately 5000 points, gradient descent provides an efficient solution to this optimization problem [Campbell and Cristianini, 1999].

Another appealing feature of SVM classification is the sparseness of its representation of the decision boundary. The location of the separating hyperplane in the feature space is specified via real-valued weights on the training set examples. Those training examples that lie far away from the hyperplane do not participate in its specification and therefore receive weights of zero. Only the training examples that lie close to the decision boundary between the two classes receive non-zero weights. These training examples are called the *support vectors*, since removing them would change the location of the separating hyperplane. The support vectors in a two-dimensional feature space are illustrated in Figure 3.

The SVM learning algorithm is defined so that, in a typical case, the number of support vectors is small compared to the total number of training examples. This property allows the SVM to classify new examples efficiently, since the majority of the training examples can be safely ignored. In essence, the SVM focuses upon the small subset of examples that are critical to differentiating between class members and non-class members, throwing out the remaining examples. This is a

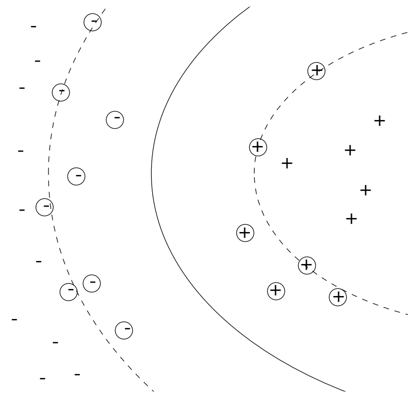


Figure 3: **A separating hyperplane in the feature space may correspond to a non-linear boundary in the input space.** The figure shows the classification boundary (solid line) in a two-dimensional input space as well as the accompanying soft margins (dotted lines). Positive and negative examples fall on opposite sides of the decision boundary. The support vectors (circled) are the points lying closest to the decision boundary.

crucial property when analyzing large data sets containing many uninformative patterns, as is the case in many data mining problems. SVMs effectively remove the uniformative patterns from the data set by assigning them weights of zero.

The maximum margin allows the SVM to select among multiple candidate hyperplanes; however, for many data sets, the SVM may not be able to find any separating hyperplane at all, either because the kernel function is inappropriate for the training data or because the data contains mislabelled examples. The latter problem can be addressed by using a *soft margin* that accepts some misclassifications of the training examples. A soft margin can be obtained in two different ways. The first is to add a constant factor to the kernel function output whenever the given input vectors are identical [Shawe-Taylor and Cristianini, 1999]. The second is to define *a priori* an upper bound on the size of the training set weights [Cortes and Vapnik, 1995]. In either case, the magnitude of the constant factor—to be added to the kernel or to bound the size of the weights—controls the number of training points that the system misclassifies. The setting of this parameter depends on the specific data at hand. Completely specifying a support vector machine therefore requires specifying two parameters: the kernel function and the magnitude of the penalty for violating the soft margin.

To summarize, a support vector machine finds a nonlinear decision function in the input space by mapping the data into a higher dimensional feature space and separating it there by means of a maximum margin hyperplane. The computational complexity of the classification operation does not depend on the dimensionality of the feature space, which can even be infinite. Overfitting is avoided by controlling the margin. The separating hyperplane is represented sparsely as a linear combination of points. The system automatically identifies a subset of informative points and uses them to represent the solution. Finally, the training algorithm solves a simple convex optimization problem. All these features make SVMs an attractive classification system. A more mathematical description of SVMs can be found in Appendix A.

4 SVMs for gene expression data

The kernel function acts as a similarity metric between examples in the training set. A simple form of similarity metric is the dot product between two vectors. Previous work [Eisen et al., 1998] has employed a normalized dot product as a similarity metric. Let X_i be the logarithm of the gene expression ratio for gene X in experimental condition i as defined in Section 2. Let the normalized feature vector, $\bar{\mathbf{X}}$ be defined as

$$\bar{X}_i = \frac{X_i}{\sqrt{\sum_{j=1}^{m} X_j^2}}, \tag{1}$$

where m is the number of elements in each expression vector. The similarity between two gene expression vectors, $\mathbf{X}$ and $\mathbf{Y}$, for the normalized dot product is defined to be $\bar{\mathbf{X}} \cdot \bar{\mathbf{Y}}$, the dot product on the normalized feature vectors. Eisen *et al.* use this measure of similiarity to perform hierarchical clustering of genes. We use essentially this same similarity metric as the kernel function in a support vector machine.

Intuitively, one drawback to support vector machine classification is that the classifier is by definition based upon a planar division of the feature space. One can easily imagine a space in which a more complex separating surface more successfully divides family members from non-family members. Through the use of an appropriate kernel function, an SVM can be constructed that produces a separating hyperplane in the feature space that corresponds to a polynomial surface in the input space. This is accomplished by raising the dot product kernel to a positive integer power. Squaring the kernel yields a convex surface in the input space. Raising the kernel to higher powers yields polynomial surfaces of higher degrees. The kernel of degree d is defined by $\left(\bar{\mathbf{X}} \cdot \bar{\mathbf{Y}} + 1\right)^d$. In the feature space of this kernel, for any gene X there are features for all d-fold interactions between mRNA measurements, represented by terms of the form $\bar{X}_{i_1}\bar{X}_{i_2}\ldots\bar{X}_{i_d}$, where $1 \leq i_1, \ldots, i_d \leq 79$. We experiment here with these kernels for degrees $d = 1, 2$ and 3, respectively, denoted below as Dot-product-1, Dot-product-2 and Dot-product-3, resp. The degree one kernel is essentially the normalized dot product kernel, and we also refer to it this way.

In a space in which the positive members of a class form one or more clusters, an accurate classifier might place a Gaussian around each cluster, thereby separating the clusters from the remaining space of non-class members. This effect can be accomplished by placing a small Gaussian over each support vector in the training set. If the width of the Gaussians is chosen well, then the sum of the support vector Gaussians will yield an accurate classifier. This is the technique used by most radial basis function classifiers [Schölkopf et al., 1997]. The formula for the Gaussian, or radial basis function, SVM kernel is

$$K(\mathbf{X}, \mathbf{Y}) = exp\left(\frac{-||\bar{\mathbf{X}} - \bar{\mathbf{Y}}||^2}{2\sigma^2}\right), \tag{2}$$

where σ is the width of the Gaussian. In our experiments, σ is set equal to the median of the Euclidean distances from each positive training set member to the nearest negative [Jaakkola et al., 1999].

The functional classes of genes examined here contain very few members relative to the total number of genes in the data set. This imbalance in the number of positive and negative training examples will cause difficulties for any classification method. For SVMs, the benefit gained by including a few class members on the correct side of the hyperplane may be exceeded by the cost associated with that hyperplane due to incorrectly labeled or inaccurately measured negative examples that also appear on the positive side of the hyperplane. In such a situation, when the magnitude of the noise in the negative examples outweighs the total number of positive examples, the optimal hyperplane located by the SVM will be uninformative, classifying all members of the training set as negative examples.

We combat this problem by modifying the matrix of kernel values computed during SVM optimization, as mentioned previously in Section 3. Let $\mathbf{K}$ be the matrix defined by the kernel function K on the training set; i.e., $\mathbf{K}_{ij} = K(\mathbf{X_i}, \mathbf{Y_i})$. By adding to the diagonal of the kernel matrix a constant whose magnitude depends upon the class of the training example, one can control the fraction of misclassified points in the two classes. This technique ensures that the positive points are not regarded as noisy labels. For positive examples, the diagonal element is given by

$$\mathbf{K}[x, x] = K(x, x) + \lambda\frac{n^+}{N}, \tag{3}$$

where n^+ is the number of positive training examples, N is the total number of training examples, and λ is a scale factor. A similar formula is used for the negative examples, with n^+ replaced by n^-. In the experiments reported here, the scale factor λ is set to 1. When the number of positive examples is small, this technique has the effect of forcing the positive examples to be relatively far from the hyperplane, whereas the negative examples can be closer. In this way, the SVM avoids the uninformative solution of classifying all positive examples as errors. This is discussed in more detail in Appendix A.

5 Methods

5.1 Expression data

All of the analyses described here are carried out using a set of 79-element gene expression vectors for 2467 genes in the budding yeast *S. cerevisiae* [Eisen et al., 1998]. The data were collected at various time points during the diauxic shift [DeRisi et al., 1997], the mitotic cell division cycle [Spellman et al., 1998a], sporulation[Chu et al., 1998], and temperature and reducing shocks. This data was used by Eisen *et al.* and is available on the Stanford web site [Eisen, 1999].

1	Tricarboxylic-acid pathway
2	Respiration chain complexes
3	Cytoplasmic ribosomal proteins
4	Proteasome
5	Histones
6	Helix-turn-helix

Table 1: **Functional classes learned by the classifiers.** Class definitions are taken from MYGD [MYGD, 1999]. The *tricarboxylic-acid pathway* is also known as the Krebs cycle. Genes in this pathway encode enzymes that break down pyruvate (produced from glucose) by oxidation. This is a key catabolic pathway to produce energy for the cell initial in the form of NADH and is also important for producing intermediates in the biosynthesis of amino acids and other compounds. The *respiration chain complexes* perform oxidation-reduction reactions that capture the energy present in NADH through electron transport and the chemiosmotic synthesis of ATP. These include the NADH deydrogenase complex, cytochrome b-c complex, and cytochrome oxidase complex, all embedded in the mitochondrial membrane. The *cytoplasmic ribosomal proteins* are a class of proteins required to make the ribosome, an RNA-protein complex in the cytoplasm encoded by mRNA. This category does not include genes for the mitochondrial ribosome. The *proteasome* consists of proteins comprising a complex responsible for general degradation of proteins and other specific protein processing events. The proteasome uses *ubiquitin*, a small peptide that marks a protein to be degraded. *Histones* interact with the DNA backbone to form nucleosomes which, with other proteins, form the chromatin of the cell. Finally, the *helix-turn-helix* class consists of genes that code for proteins that contain the helix-turn-helix structural motif. This does not constitute a functional class, and is included only as a control.

5.2 Biological functional classes

Classification accuracy is tested using six functional classes defined by the MIPS Yeast Genome Database [MYGD, 1999] (see Table 1). MYGD class definitions come from biochemical and genetic studies of gene funtion, while the microarry expression data measures mRNA levels of genes. Therefore, many classes in MYGD, such as broad structural classes such as the protein kinases, will not be predictable only by examining expression data. The first five classes in Table 1 are selected because they represent categories of genes that are expected, on biological grounds, to exhibit similar expression profiles. Furthermore, Eisen *et al.* showed that these classes cluster well using hierarchical clustering based upon the normalized dot product [Eisen et al., 1998]. The sixth class, the helix-turn-helix proteins, is included as a control group. Since there is no reason to believe that the members of this class are similarly regulated, we do not expect any classifier to learn to recognize members of this class based upon mRNA expression measurements.

5.3 Experimental setup

Performance is measured using a three-way cross-validated experiment. The gene expression vectors are randomly divided into three groups. Classifiers are then trained using two groups and tested on the third.

The performance of each classifier is measured by examining how well the classifier identifies the positive and negative examples in the test set. Most of the classification methods return a rank ordering of the test set. Given this ordering and a classification threshold, each gene in the test set can be labeled in one of four ways: false positives are genes that the classifier places within the given class, but MYGD classifies as non-members; false negatives are genes that the classifier places outside the class, but MYGD classifies as members; true positives are class members according to both the classifier and MYGD, and true negatives are non-members according to both. For each method, we find the classification threshold that minimizes the cost function, $fp + 2 \cdot fn$, where fp is the number of false positives, and fn is the number of false negatives. The false negatives are weighted more heavily than the false positives because, for these data, the number of positive examples is small compared to the number of negatives. Results are reported in terms of the false positive and false negative error rates as well as the cost at the minimal classification threshold.

Note that the two decision tree methods do not produce a rank ordering of test set points, making it impossible to vary the classification threshold. Therefore, for the decision tree methods we use the default threshold, rather than the one found by minimizing the cost function.

5.4 Support vector machines

Because SVM learning is guaranteed to converge to a single global solution, the algorithm itself is fairly simple. Our implementation follows the formulation of [Jaakkola et al., 1998]. This approach differs slightly from that of [Vapnik, 1998], although the geometric interpretation remains the same. Let $\mathbf{X} = \mathbf{X_1} \ldots \mathbf{X_n}$ be a set of training examples, and $y = y_1 \ldots y_n$ be the corresponding set of classifications, where $y_i = 1$ if $\mathbf{X_i}$ is a member of the class to be learned, and $y_i = -1$

otherwise. Define the discriminant function

$$L(\mathbf{X}) = \sum_{i=1}^{n} y_i \alpha_i K(\mathbf{X}, \mathbf{X_i}), \tag{4}$$

where α_i is the weight of training example $\mathbf{X_i}$. The goal is to learn a set of weights that maximize the following objective function:

$$\begin{aligned} J(\alpha) &= \sum_{i=1}^{n} \alpha_i (2 - y_i L(\mathbf{X_i})) & (5) \\ &= 2\sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{X_i}, \mathbf{X_j}) & (6) \end{aligned}$$

This maximum can be obtained by iteratively updating the weights using the following update rule:

$$\alpha_i \leftarrow f\left(\frac{1 - y_i L(\mathbf{X_i}) + \alpha_i K(\mathbf{X_i}, \mathbf{X})}{K(\mathbf{X_i}, \mathbf{X})}\right), \tag{7}$$

where $f(x) = x$ for $x > 0$ and $f(x) = 0$ for $x <= 0$. Note that equation 7 differs from the corresponding equation in [Jaakkola et al., 1998], in that the weights α_i are not constrained to be less than 1. This difference arises because we implement the soft margin by modifying the diagonal of the kernel matrix, rather than by truncating the weights.

The output of the SVM learning algorithm is the optimized set of weights $\alpha_1 \ldots \alpha_n$. The class of a new input vector $\mathbf{X}$ is then given by the sign of the discriminant $L(\mathbf{X})$ computed using the optimized weights.[1]

5.5 Decision trees

We compare the performance of the SVM classifiers described above with that of four other classifiers. The first two are decision tree classifiers. Decision trees are a standard tool in data mining, and many are available in packages such as CART [Breiman et al., 1984] and C4.5 [Quinlan, 1997]. Decision trees are generally preferred over other nonparametric techniques because of the readability of their learned hypotheses and the efficiency of training and evaluation.

Decision trees are rooted, usually binary trees, with simple classifiers placed at each internal node and a classification at each leaf. In order to evaluate a particular tree T with respect to an input x, each classifier in the tree is assigned the argument x. The outputs of the simple classifiers at the nodes determine a unique path from the root to a leaf of the decision tree: at each internal node, the left edge to a child is taken if the output of the function associated with that internal node is +1, and vice versa if it is -1. This path is known as the *evaluation path*. The value of the function $T(x)$ is the classification associated with the leaf reached by the evaluation path.

Decision trees are generally learned by means of a top down growth procedure, which starts from the root node and greedily chooses a split of the data that maximizes some cost function, usually a measure of the "impurity" of the subsamples implicitly defined by the split. After choosing a split, the subsamples are then mapped to the two children nodes. This procedure is then recursively

[1] The scaled dot product kernel gives better performance using a threshold that is optimized on the training set, so we report results for this threshold, rather than a threshold of 0.

applied to the children, and the tree is grown until some stopping criterion is met. The tree is then used as a starting point for a bottom up search, performing a pruning of the tree. This eliminates nodes that are redundant or are unable to "pay for themselves" in terms of the cost function.

Typically, the simple classifier at an internal node compares one of the input attributes against a threshold. This test partitions the input space with axis parallel splits. The standard algorithm of this kind is C4.5 [Quinlan, 1997]. Another strategy uses hyperplanes in general position. This is the technique adopted by systems like OC1. We use an improved version of OC1, called MOC1, which implements a bias toward large margin splits in the purity measure, theoretically motivated by Vapnik-Chervonenkis theory. MOC1 has been shown to outperform the standard OC1 [Wu et al., 1999].

We use the systems C4.5 and MOC1 in their basic version with all the default settings. Note that it would be possible to devise modifications of the same systems to account, for example, for the unequal numbers of positive and negative training examples, which might improve their performance.

5.6 Parzen windows

Parzen windows classification is a technique for nonparametric density estimation, which can also be used for classification. Using a given kernel function, the technique approximates a given training set distribution via a linear combination of kernels centered on the observed points. In this work, we separately approximate densities for each of the two classes, and we assign a test point to the class with maximal posterior probability.

The resulting algorithm is extremely simple and closely related to support vector machines. The decision function is

$$f(\mathbf{X}) = sign(\sum y_i K(\mathbf{X}_i, \mathbf{X})), \tag{8}$$

where the kernel function K is the radial basis function of Equation 2, without normalization applied to the inputs. As for the radial basis SVM, a constant is added to the kernel function whenever the two inputs are identical (Equation 3).

The Parzen windows classification algorithm does not require any training phase; however, the lack of sparseness makes the test phase quite slow. Furthermore, although asymptotical convergence guarantees on the perfomance of Parzen windows classifiers exist [Duda and Hart, 1973], no such guarantees exist for finite sample sizes.

Parzen windows can be regarded as a generalization of k-nearest neighbor techniques. Rather than choosing the k nearest neighbors of a test point and labelling the test point with the weighted majority of its neighbors' votes, one can consider all points in the voting scheme and assign their weight by means of the kernel function. With Gaussian kernels, the weight decreases exponentially with the square of the distance, so far away points are practically irrelevant. The width σ of the Guassian determines the relative weighting of near and far points. Tuning this parameter controls the predictive power of the system. We have empirically optimized the value of σ.

5.7 Fisher's linear discriminant

Fisher's linear discriminant is a classification method that projects high-dimensional data onto a line and performs classification in this one-dimensional space. The projection maximizes the distance between the means of the two classes while minimizing the variance within each class. This defines the Fisher criterion, which is maximized over all linear projections, w:

$$J(w) = \frac{|m_1 - m_2|^2}{s_1^2 + s_2^2} \tag{9}$$

where m represents a mean, s^2 represents a variance, and the subscripts denote the two classes. In signal theory, this criterion is also known as the signal-to-interference ratio. Maximizing this criterion yields a closed form solution that involves the inverse of a covariance-like matrix. This method has strong parallels to linear perceptrons. We learn the threshold by optimizing a cost function on the training set.

6 Results and discussion

Our experiments show the benefits of classifying genes using support vector machines trained on DNA microarray expression data. We begin with a comparison of SVMs versus four non-SVM methods and show that SVMs provide superior performance. We then examine more closely the performance of several different SVMs and demonstrate the superiority of the radial basis function SVM. Finally, we examine in detail some of the apparent errors made by the radial basis function SVM and show that many of the apparent errors are in fact biologically reasonable classifications. Most of the results reported here can be accessed via the web at http://www.cse.ucsc.edu/research/compbio.

For the data analyzed here, support vector machines provide better classification performance than the competing classifiers. Tables 2 and 3 summarize the results of a three-fold cross-validation experiment using all eight of the classifiers described in Section 5, including four SVM variants, Parzen windows, Fisher's linear discriminant and two decision tree learners. The five columns labeled "Learned threshold" summarize classification performance. In this case, the method must produce a binary classification label for each member of the test set. Overall performance of each method is judged using the cost function, $fp + (2 \cdot fn)$. For every class (except the last, unlearnable class), the best-performing method using the learned threshold is the radial basis support vector machine. Other cost functions, with different relative weights of the false positive and false negative rates, yield similar rankings of performance. These results are not statistically sufficient to demonstrate unequivocally that one method is better than the other; however, they do give some evidence. For example, in five separate tests, the radial basis SVM performs better than Fisher's linear discriminant. Under the null hypothesis that the methods are equally good, the probability that the radial basis SVM would be the best all five times is $1/32 = 0.03125$.

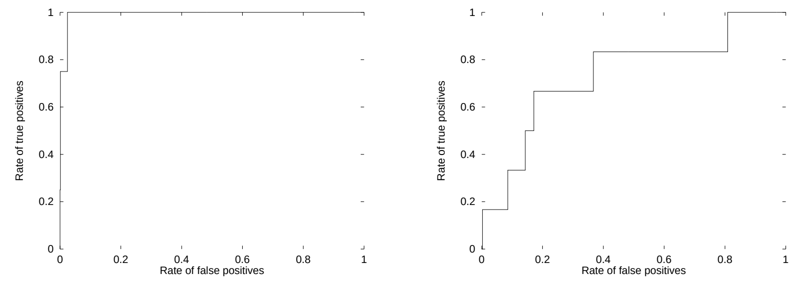
In addition to producing binary classification labels, six of the eight methods produce a ranked list of the test set examples. This ranked list provides more information than the simple binary classification labels. For example, scanning the ranked lists allows the experimenter to easily focus on the genes that lie on the border of the given class. Ranked lists produced by the radial basis SVM for each of the five classes are available at http://www.cse.ucsc.edu/research/compbio/-genex. A perfect classifier will place all positive test set examples before the negative examples in the ranked list and will correctly specify the decision boundary to lie between the positives and the negatives. An imperfect classifier, on the other hand, will either produce an incorrect ordering of the test set examples or use an inaccurate classification threshold. Thus, the performance can be improved by fixing either the ranking or the threshold. However, given an improper ranking,

		Learned threshold					Optimized threshold				
Class	Method	FP	FN	TP	TN	Cost	FP	FN	TP	TN	Cost
Tricarboxylic acid	Radial SVM	8	8	9	2442	24	4	7	10	2446	18
	Dot-product-1 SVM	11	9	8	2439	29	3	6	11	2447	15
	Dot-product-2 SVM	5	10	7	2445	25	4	6	11	2446	16
	Dot-product-3 SVM	4	12	5	2446	28	4	6	11	2446	16
	Parzen	4	12	5	2446	28	0	12	5	2450	24
	FLD	9	10	7	2441	29	7	8	9	2443	23
	C4.5	7	17	0	2443	41	–	–	–	–	–
	MOC1	3	16	1	2446	35	–	–	–	–	–
Respiration	Radial SVM	9	6	24	2428	21	8	4	26	2429	16
	Dot-product-1 SVM	21	10	20	2416	41	6	9	21	2431	24
	Dot-product-2 SVM	7	14	16	2430	35	7	6	24	2430	19
	Dot-product-3 SVM	3	15	15	2434	33	7	6	24	2430	19
	Parzen	22	10	20	2415	42	7	12	18	2430	31
	FLD	10	10	20	2427	30	14	4	26	2423	22
	C4.5	18	17	13	2419	52	–	–	–	–	–
	MOC1	12	26	4	2425	64	–	–	–	–	–
Ribosome	Radial SVM	9	4	117	2337	17	6	1	120	2340	8
	Dot-product-1 SVM	13	6	115	2333	25	11	1	120	2335	13
	Dot-product-2 SVM	7	10	111	2339	27	9	1	120	2337	11
	Dot-product-3 SVM	3	18	103	2343	39	7	1	120	2339	9
	Parzen	6	8	113	2340	22	5	8	113	2341	21
	FLD	15	5	116	2331	25	8	3	118	2338	14
	C4.5	31	21	100	2315	73	–	–	–	–	–
	MOC1	26	26	95	2320	78	–	–	–	–	–

Table 2: **Comparison of error rates for various classification methods.** Classes are as described in Table 1. The methods are the radial basis function SVM, the SVMs using the scaled dot product kernel raised to the first, second and third power, Parzen windows, Fisher's linear discriminant, and the two decision tree learners, C4.5 and MOC1. The next five columns are the false positive, false negative, true positive and true negative rates summed over three cross-validation splits, followed by the cost, which is the number of false positives plus twice the number of false negatives. These five columns are repeated twice, first using the threshold learned from the training set, and then using the threshold that minimizes the cost on the test set. The threshold optimization is not possible for the decision tree methods, since they do not produce ranked results.

Class	Method	Learned threshold					Optimized threshold				
		FP	FN	TP	TN	Cost	FP	FN	TP	TN	Cost
Proteasome	Radial SVM	3	7	28	2429	17	4	5	30	2428	14
	Dot-product-1 SVM	14	11	24	2418	36	2	7	28	2430	16
	Dot-product-2 SVM	4	13	22	2428	30	4	6	29	2428	16
	Dot-product-3 SVM	3	18	17	2429	39	2	7	28	2430	16
	Parzen	21	5	30	2411	31	3	9	26	2429	21
	FLD	7	12	23	2425	31	12	7	28	2420	26
	C4.5	17	10	25	2415	37	–	–	–	–	–
	MOC1	10	17	18	2422	44	–	–	–	–	–
Histone	Radial SVM	0	2	9	2456	4	0	2	9	2456	4
	Dot-product-1 SVM	0	4	7	2456	8	0	2	9	2456	4
	Dot-product-2 SVM	0	5	6	2456	10	0	2	9	2456	4
	Dot-product-3 SVM	0	8	3	2456	16	0	2	9	2456	4
	Parzen	2	3	8	2454	8	1	3	8	2455	7
	FLD	0	3	8	2456	6	2	1	10	2454	4
	C4.5	2	2	9	2454	6	–	–	–	–	–
	MOC1	2	5	6	2454	12	–	–	–	–	–
Helix-turn-helix	Radial SVM	1	16	0	2450	33	0	16	0	2451	32
	Dot-product-1 SVM	20	16	0	2431	52	0	16	0	2451	32
	Dot-product-2 SVM	4	16	0	2447	36	0	16	0	2451	32
	Dot-product-3 SVM	1	16	0	2450	33	0	16	0	2451	32
	Parzen	14	16	0	2437	46	0	16	0	2451	32
	FLD	14	16	0	2437	46	0	16	0	2451	32
	C4.5	2	16	0	2449	34	–	–	–	–	–
	MOC1	6	16	0	2445	38	–	–	–	–	–

Table 3: **Comparison of error rates for various classification methods (continued).** See caption for Table 2.



(a) Tricarboxylic-acid pathway (b) Helix-turn-helix

Figure 4: **Receiver operating characteristic curves for a learnable and non-learnable class.** Each curve plots the rate of true positives as a function of the rate of false positives for varying classification thresholds. Both curves were generated by training a radial basis SVM on two-thirds of the data and testing on the remaining one-third.

no classification threshold can yield perfect performance. Therefore, we focus on finding a correct ranking of the test set. The columns labeled "Optimized threshold" in Tables 2 and 3 show the best performance that could be achieved if the classifier were capable of learning the decision threshold perfectly. These results further demonstrate the superior performance of the radial basis SVM: it performs best in four out of five of the learnable classes. Furthermore, the performance of the scaled dot product SVMs improves so that in nearly every class, the best four classifiers are the four SVM methods.

As expected, the results also show the inability of these classifiers to learn to recognize the class of genes that produce helix-turn-helix (HTH) proteins. Since helix-turn-helix proteins are not expected to share similar expression profiles, we do not expect any classifier to be capable of learning to recognize this class from gene expression data. Most methods uniformly classify all test set sequences as non-HTHs. The unlearnability of this class is also apparent from a receiver operating characteristic (ROC) analysis of the classification results. Figure 4 shows two ROC curves, which plot the rate of true positives as a function of the rate of false positives as the classification threshold is varied. For a learnable class, such as the genes participating in the tricarboxylic-acid pathway, the false positive sequences cluster close together with respect to the classification threshold. For the HTHs, by contrast, the classification threshold must be varied widely in order to classify all class members as positives. Since the positive class members are essentially random with respect to the classification threshold, the ROC curve shows clearly that this gene class is unlearnable and hence unlikely to be co-regulated.

In addition to demonstrating the superior performance of SVMs relative to non-SVM methods, the results in Tables 2 and 3 indicate that the radial basis SVM performs better than SVMs that use a scaled dot product kernel. In order to verify this difference in performance, we repeated the three-fold cross-validation experiment four more times, using four different random splits of the data. Table 4 summarizes the cost for each SVM on each of the five random splits. The total cost in all five experiments is reported in the final column of the table. The radial basis SVM performs better than the scaled dot product SVMs for all classes except the histones, for which all four methods perform identically. Again, this is not conclusive evidence that the radial basis SVM is superior to the other methods, but it is suggestive.

Class	Kernel	Cost for each split					Total
Tricarboxylic acid	Radial	18	21	15	22	21	97
	Dot-product-1	15	22	18	23	22	100
	Dot-product-2	16	22	17	22	22	99
	Dot-product-3	16	22	17	23	22	100
Respiration	Radial	16	18	23	20	16	93
	Dot-product-1	24	24	29	27	23	127
	Dot-product-2	19	19	26	24	23	111
	Dot-product-3	19	19	26	22	21	107
Ribosome	Radial	8	12	15	11	13	59
	Dot-product-1	13	18	14	16	16	77
	Dot-product-2	11	16	14	16	15	72
	Dot-product-3	9	15	11	15	15	65
Proteasome	Radial	14	10	9	11	11	55
	Dot-product-1	16	12	12	17	19	76
	Dot-product-2	16	13	15	17	17	78
	Dot-product-3	16	13	16	16	17	79
Histone	Radial	4	4	4	4	4	20
	Dot-product-1	4	4	4	4	4	20
	Dot-product-2	4	4	4	4	4	20
	Dot-product-3	4	4	4	4	4	20

Table 4: **Comparison of SVM performance using various kernels.** For each of the MYGD classifications, SVMs were trained using four different kernel functions on five different random three-fold splits of the data, training on two-thirds and testing on the remaining third. The first column contains the class, as described in Table 1. The second column contains the kernel function, as described in Table 2. The next five columns contain the threshold-optimized cost (i.e., the number of false positives plus twice the number of false negatives) for each of the five random three-fold splits. The final column is the total cost across all five splits.

Class	Number of splits 1	2	3	4	5
Tricarboxylic-acid pathway	7	2	2	1	8
Respiration chain complexes	9	1	2	4	6
Cytoplasmic ribosomes	5	2	3	2	4
Proteasome	6	0	1	0	5
Histones	0	0	0	0	2

Table 5: **Consistency of errors across five different random splits of the data.** For each of the MYGD classifications listed in the first column, radial basis SVMs were trained on five different random three-fold splits of the data, training on two-thirds and testing on the remaining third. An entry in column n of the table represents the total number of genes misclassified with respect to the MYGD classification in n of the five random splits. Thus, for example, eight genes were consistently mislabeled by the SVMs trained on genes from the tricarboxylic-acid pathway.

Besides providing improved support for the claim that the radial basis SVM outperforms the scaled dot product SVMs, repeating the three-fold cross-validation experiment also provides insight into the consistency with which the SVM makes mistakes. A classification error may occur because the MYGD classification actually contains an error; on the other hand, some classification errors may arise simply because the gene is a borderline case, and may or may not appear as an error, depending on how the data is randomly split into thirds. Table 5 summarizes the number of errors that occur consistently throughout the five different experiments. The second column lists the number of genes that a radial basis SVM misclassifies only once in the five experiments. The right-most column lists the number of genes that are consistently misclassified in all five experiments. These latter genes are of much more interest, since their misclassification cannot be attributed to an unlucky split of the data.

Table 6 lists the 25 genes referred to in the final column of Table 5. These are genes for which the radial basis support vector machine consistently disagrees with the MYGD classification. Many of these disagreements reflect the different perspective provided by the expression data concerning the relationships between genes. The microarray expression data represents the genetic response of the cell to various environmental perturbations, whereas the SVM classifies genes based on how similar their expression pattern is to genes of known function. The MYGD definitions of functional classes have been arrived at through biochemical experiments that classify gene products by what they do, not how they are regulated. These different perspectives sometimes lead to different functional classifications. For example, in MYGD the members of a complex are defined as what copurifies with the complex, whereas in the expression data a complex is defined by what genes need to be transcribed for proper functioning of the complex. The above example will lead to disagreements between the SVM and MYGD in the form of false positives. Disagreements between the SVM and MYGD in the form of false negatives occur for a number of reasons. First, genes that are classified in MYGD primarily by structure (e.g., protein kinases) may not be similarly classified by the SVM. Second, genes that are regulated at the translational level or protein level, rather than at the trancriptional level measured by the microarray experiments, cannot be correctly classified by expression data alone. Third, genes for which the microarray data is corrupt cannot

Family	Gene	Locus	Error	Description
TCA	YPR001W	CIT3	FN	mitochondrial citrate synthase
	YOR142W	LSC1	FN	α subunit of succinyl-CoA ligase
	YNR001C	CIT1	FN	mitochondrial citrate synthase
	YLR174W	IDP2	FN	isocitrate dehydrogenase
	YIL125W	KGD1	FN	α-ketoglutarate dehydrogenase
	YDR148C	KGD2	FN	component of α-ketoglutarate dehydrogenase complex in mitochondria
	YDL066W	IDP1	FN	mitochondrial form of isocitrate dehydrogenase
	YBL015W	ACH1	FP	acetyl CoA hydrolase
Resp	YPR191W	QCR2	FN	ubiquinol cytochrome-c reductase core protein 2
	YPL271W	ATP15	FN	ATP synthase epsilon subunit
	YPL262W	FUM1	FP	fumarase
	YML120C	NDI1	FP	mitochondrial NADH ubiquinone 6 oxidoreductase
	YKL085W	MDH1	FP	mitochondrial malate dehydrogenase
	YDL067C	COX9	FN	subunit VIIa of cytochrome c oxidase
Ribo	YPL037C	EGD1	FP	β subunit of the nascent-polypeptide-associated complex (NAC)
	YLR406C	RPL31B	FN	ribosomal protein L31B (L34B) (YL28)
	YLR075W	RPL10	FP	ribosomal protein L10
	YAL003W	EFB1	FP	translation elongation factor EF-1β
Prot	YHR027C	RPN1	FN	subunit of 26S proteasome (PA700 subunit)
	YGR270W	YTA7	FN	member of CDC48/PAS1/SEC18 family of ATPases
	YGR048W	UFD1	FP	ubiquitin fusion degradation protein
	YDR069C	DOA4	FN	ubiquitin isopeptidase
	YDL020C	RPN4	FN	involved in ubiquitin degradation pathway
Hist	YOL012C	HTA3	FN	histone-related protein
	YKL049C	CSE4	FN	required for proper kinetochore function

Table 6: **Consistently misclassified genes.** The table lists all 25 genes that are consistently misclassified by SVMs trained using the MYGD classifications listed in Table 1. Two types of errors are included: a false positive (FP) occurs when the SVM includes the gene in the given class but the MYGD classification does not; a false negative (FN) occurs when the SVM does not include the gene in the given class but the MYGD classification does.

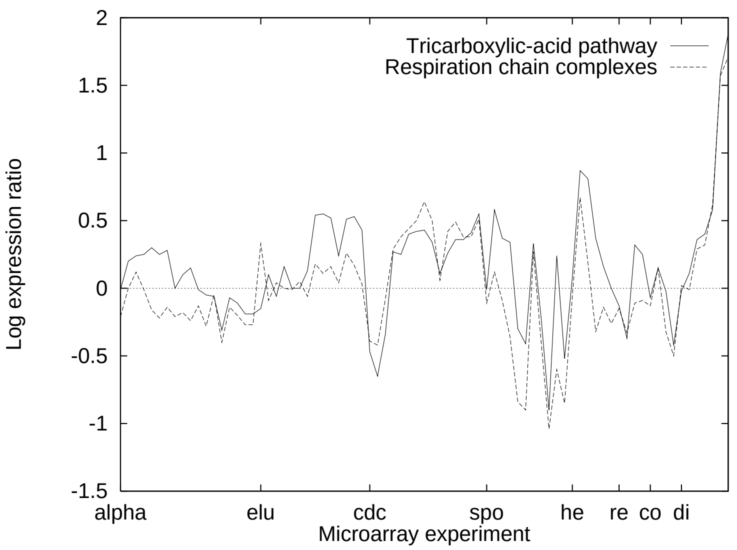


Figure 5: **Similarity between the average expression profiles of the tricarboxylic-acid pathway and respiration chain complexes.** Each series represents the average log expression ratio for all genes in the given family plotted as a function of DNA microarray experiment. Ticks along the X-axis represent the beginnings of experimental series, as described in Figure 1.

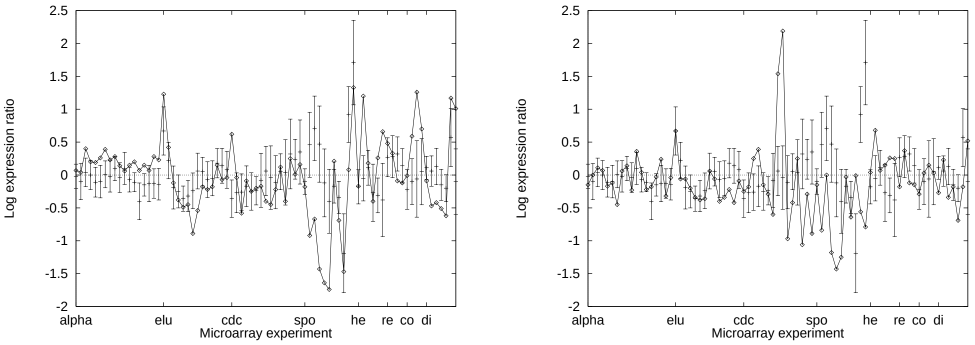
be correctly classified. Disagreements represent the cases where the different perspectives of the SVM and MYGD lead to different functional classifications and illustrate the new information that expression data brings to biology.

6.1 False positives

Many of the false positives in Table 6 are known from biochemical studies to be important for the same functional class assigned by the SVM, even though MYGD has not included these genes in their functional class. For example YAL003W, a false positive assigned repeatedly to the cytoplasmic ribosome class, is not strictly a ribosomal protein. However, it does encode a translation elongation factor, EFB1, known to be required for the proper functioning of the ribosome [Kinzy and J. L. Woolford, 1995]. The cell ensures that expression of this factor keeps pace with the expression of ribosomal proteins. Thus, the SVM classifies YAL003W with ribosomal proteins.

The respiration complexes class provides another example, YML120C, NADH:ubiquinone oxidoreductase. In yeast, YML120C replaces respiration complex 1 [Marres et al., 1991], and while it does not pump protons across the mitochondrial inner membrane, this gene is crucial to the proper functioning of the respiration complexes. Without YML120C, the respiration chain is unable to transfer high energy electrons from NADH to ubiquinone, and respiration stops [Marres et al., 1991, Kitajima-Ihara and Yagi, 1998]. In the proteasome class YGR048W, ufd1, is classified by the SVM as part of the proteasome class. While YGR048W is not strictly part of the proteasome, it is necessary for proper functioning of the ubiquitin degradation pathway [Johnson et al., 1995], which delivers proteins to the proteasome for proteolysis.

Other examples include the classification of members of the tricarboxylic-acid (TCA) pathway,



(a) YDL020C (b) YDR069C

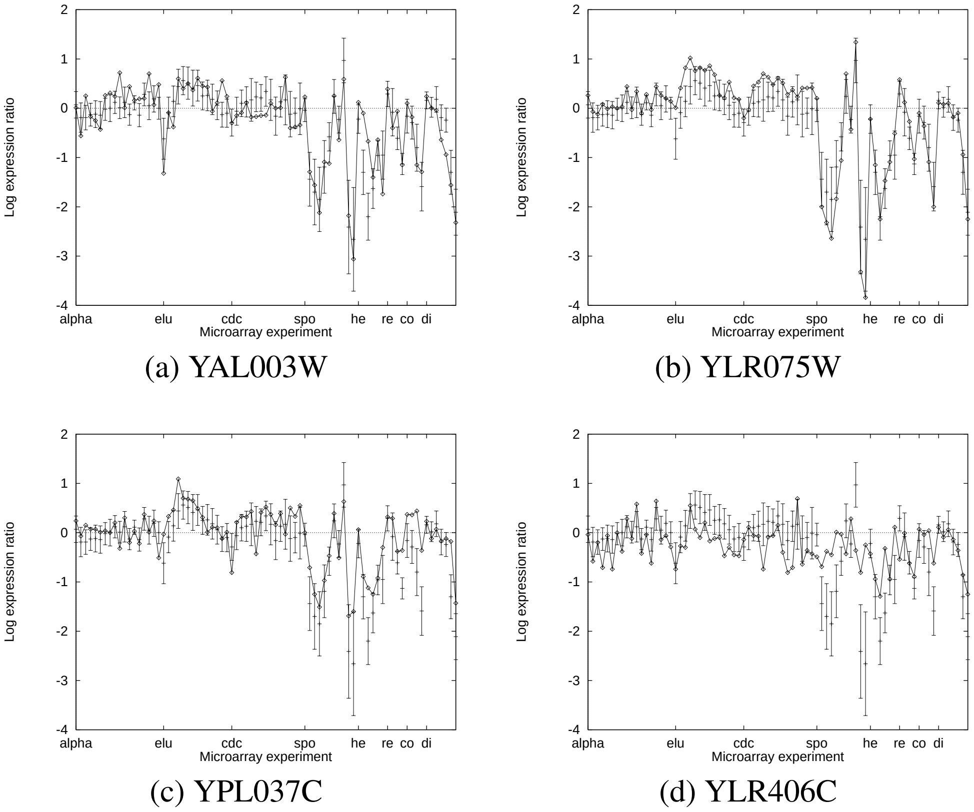
Figure 6: **Expression profiles of two false negative genes for the proteasome class.** Each figure shows the expression profile for a single gene, along with standard deviation bars for the proteasome class. Ticks along the X-axis represent the beginnings of experimental series, as described in Figure 1.

YPL262W and YKL085W, as members of the respiration chain complexes. While MYGD separates the tricarboxylic-acid (TCA) pathway and the respiration chain complexes, these two classes are known to be tightly coupled in the production of ATP. This relationship is represented in the expression data by the similarity between the expression profiles of the two classes, as shown in Figure 5 and leads the SVM to classify YPL262W and YKL085W as respiration complexes. Thus, while MYGD considers these two classes separate, both the expression data and other experimental work suggest that YPL262W and YKL085W have important roles to play in the function of the respiration complexes.

6.2 False negatives

Some of the false negatives produced by the support vector machine occur when a protein that was assigned to a functional class in MYGD based on structural similarity has a special function that demands a different regulation strategy. For example, YKL049C is classified as a histone protein by MYGD based on its 65% amino acid similarity with histone protein H3. YKL049C is thought to act as part of the centromere [Stoler et al., 1995], and while it is related to histones, the expression data shows that it is not coregulated with other histone genes. Therefore, the SVM does not assign YLK049C to the histone class. A similar situation arises in the proteasome class. Both YDL020C and YDR069C are physically associated with the proteasome [Fujimuro et al., 1998, Papa et al., 1999]. However, these proteins are not intrinsic subunits of the proteasome, but are loosely associated auxiliary factors [Glickman et al., 1998, Papa et al., 1999]. The SVM does not classify them as belonging to the proteasome because they are regulated differently from the rest of the proteasome during sporulation, as shown in Figure 6.

One limitation inherent in the use of gene expression data to identify genes that function together is that some genes are regulated primarily at the translational and protein levels. For example, six of the seven cases in which the SVM was unable to assign members of the TCA class are genes encoding citrate synthase, isocitrate dehydrogenase of α-ketoglutarate dehydrogenase. The enzymatic activities of these proteins are known to be regulated allosterically by ADP/ATP,



(a) YAL003W

(b) YLR075W

(c) YPL037C

(d) YLR406C

Figure 7: **Expression profiles of four genes consistently mis-classified with respect to the MYGD class of cytoplasmic ribosomal proteins.** Each figure shows the expression profile for a single gene, along with standard deviation bars for the class of cytoplasmic ribosomal proteins. Ticks along the X-axis represent the beginnings of experimental series, as described in Figure 1. The genes in Figures (a)-(c) are false positives; i.e., the SVM places them in the class but MYGD does not. The gene in Figure (d) is a false negative.

succinyl-CoA, and NAD+/NADPH [Garrett and Grisham, 1995, pp. 619–622]. These enzymes are regulated primarily by means that do not involve changes in mRNA level. Thus, the SVM will not be able to correctly classify them by expression data alone.

Other discrepancies appear to be caused by corrupt data. For example, the SVM classifies YLR075W as a cytoplasmic ribosomal protein, but MYGD does not. YLR075W is in fact a ribosomal protein [Wool et al., 1995, Dick et al., 1997]. The similarity between the YLR075W expression profile and the profile of the cytoplasmic ribosomal proteins is evident in Figure 7(b). This discrepency is an oversight in MYGD, which has since been corrected [Mannhaupt, 1999]. Other errors occur in the expression data itself. Occasionally, the microarrays contain bad probes or are damaged, and some locations in the gene expression matrix are marked as containing corrupt data. Three of the genes listed in Table 6 (YDL067C, YOR142W and YHR027C) are marked as such [Eisen, 1999]. In addition, although the SVM correctly assigns YDL075W to the ribosomal protein class, YLR406C, essentially a duplicate copy of YDL075W is not assigned to that class. The microarrays are not sensitive enough to differentiate between two such similar genes; there-

Gene	Weight	Errors
YLR075W	2.093	5
YOR276W	1.016	4
YNL209W	0.977	4
YAL003W	0.930	5
YPL037C	0.833	5
YKR059W	0.815	2
YML106W	0.791	1
YDR385W	0.771	2
YPR187W	0.767	1
YJL138C	0.757	3

Table 7: **The magnitude of the training set weights predicts outliers.** The average weight of each gene was computed across five three-fold cross-validation tests of the radial basis SVM trained on the cytoplasmic ribosomes, and the genes were ranked accordingly. The table shows the ten negative examples with the largest weights, their average weights, and the total number of times (out of five) that each gene was misclassified.

fore, it is likely that the YLR406C data is also questionable. The profile for this gene is shown in Figure 7(d).

No immediate explanation is available for the discrepancies involving the remaining six genes. These genes include one false positive TCA (YBL015W), two false negative respiration chain complexes (YPR191W and YPL271W), a false positive cytoplasmic ribosomal protein (YPL037C—see Figure 7(c)), a false negative proteasome (YGR270W), and a false negative histone (YOL012C). Further experiments would be required to determine whether these misclassifications are artifacts or are clues to the genuine biological role of these proteins.

The misclassified genes described in Table 6 were found by classifying the data using trained SVMs and identifying errors. However, many of these outlier genes could have been identified during the training phase. Genes that are misclassified in the training set are likely to be outliers with respect to their labeled class. Consequently, these genes will violate the soft margin of the SVM and will hence receive large weights (α_i in the formulation of Section 5.4). Table 7 shows the ten largest average weights for negative training set examples from the cytoplasmic ribosome class. As expected, these examples are the ones most often misclassified by the trained SVMs. The information in Table 7 could have been used to perform data cleaning, automatically removing inaccurate classifications from the training set [Guyon et al., 1996]. Such a procedure would have removed from the training data the mislabeled gene YLR075W.

7 Conclusions and future work

We have demonstrated that support vector machines can accurately classify genes into functional categories based upon expression data from DNA microarray hybridization experiments. Among the techniques that we examined, the SVM that uses a radial basis kernel function provides the best performance—better than Parzen windows, Fisher's linear discriminant, two decision tree classifiers, and three other SVMs that use a scaled dot product kernel. These results were generated in a supervised fashion, as opposed to the unsupervised clustering algorithms that have been previously considered [Eisen et al., 1998, Tamayo et al., 1999]. The supervised learning framework allows a researcher to start with a set of interesting genes and ask two questions: *What other genes are related to my set*, and *Does my set contain genes that do not belong*? Furthermore, the support vectors identified by the SVM effectively define the boundary of the training set of genes. This ability to focus on the few informative genes out of the vast landscape of uninformative genes is fundamental to making scientific insight.

The experiments reported here were performed using only expression data for genes that already have functional annotation. The expression data for the remaining *S. cerevisiae* genes are not currently available for all experimental conditions. If the data were available, the SVMs produced here would undoubtedly identify among the unannotated genes additional members of the five MYGD classes.

One significant benefit offered by SVMs is scalability. The number of support vectors selected by the SVM learning algorithm is usually small, even for very large training sets, and the resulting SVM is consequently an efficient classifier. In this work, training a radial basis SVM using two-thirds of the data set (1645 examples) takes an average of 89.5 CPU seconds on a DEC Alpha 4100 workstation running at 466 MHz. The resulting machine contains only 216 support vectors on average. Thus, classifying a new gene requires comparisons with only approximately 13.1% of the training set.

Scalability is essential because the amount of available gene expression data will soon increase dramatically. We will have larger training sets that include more genes and more detailed expression profiles. When hundreds, and soon thousands, of mRNA expressions measurements under different conditions become available for each gene, each measurement will still, by itself, give only partial and inconclusive information about any given functional classification of the gene. However, all these different mRNA measurements taken together may often provide enough information to classify the gene with very high confidence. This is much like the process whereby many observations of the same underlying signal plus independent noise can, via the central limit theorem, be reduced to one highly reliable observation of the underlying signal.

In addition to large quantities of mRNA expression data, SVMs are capable of using data about genes from other sources. Our current work uses only DNA microarray expression data, but similar SVMs could be constructed using other gene features, such as the presence of transcription factor binding sites in the promoter region or sequence features of the translated protein, e.g. as in [Jaakkola et al., 1999]. We have begun working with SVMs that classify using training vectors concatenated from multiple sources using the methods from [Jaakkola et al., 1999, Jaakkola and Haussler, 1998].

We have described some of the issues involved in selecting an appropriate kernel function. Although the simple radial basis kernel function provides excellent performance, a better SVM could be constructed that incorporates prior knowledge about the classification domain. One avenue for such research involves kernels that explicitly account for dependencies among elements in the expression profiles. Such dependencies would arise, for example, in a data set constructed from a series of microarray experiments with sufficiently small time steps between samples. In this context, we have experimented with a modified version of the dot product kernel that interposes an inverted covariance matrix betwen the two expression vectors. This kernel yields performance that is intermediate between the dot product kernel and the radial basis kernel; however, computing and inverting the covariance matrix is computationally expensive.

Any supervised learning algorithm requires a gold standard classification that will function as the teacher signal. Here, we have used MYGD classifications as our gold standard. The MYGD classifications are undoubtedly incomplete and may be biased, but they are the best classifications available, given the currently limited knowledge about functional classes and how those classes are reflected in microarray expression data. We have shown that SVMs can learn to recognize functional classes even from a noisy teacher signal provided by the MYGD classifications. Furthermore, the magnitudes of the optimized weights, as well as the discriminant values of the training set, provide accurate indicators of outliers in the training set that are likely to have been misclassified by the teacher signal. The ability to identify outliers suggests a bootstrapping approach, in which an initially noisy gold standard classification is refined by the SVM [Guyon et al., 1996]. This bootstrapping method could be applied to classifications learned via unsupervised methods, such as those of [Eisen et al., 1998] and [Tamayo et al., 1999].

Similarity and distance metrics play a fundamental role in both supervised and unsupervised methods for the analysis of mRNA expression data and for other pattern recognition problems. For example, Eisen *et al.* cluster mRNA expression vectors using hierarchical agglomerative clustering with a Pearson correlation coefficient similarity metric. Tamayo *et al.* cluster gene expression data using self-organizing maps [Tamayo et al., 1999], which rely on a distance metric.

Any similarity metric defined by kernel function $K(\mathbf{X}, \mathbf{Y})$, or equivalently, any positive definite function, can be converted into a distance function $d(\mathbf{X}, \mathbf{Y})$ via the equation $d^2(\mathbf{X}, \mathbf{Y}) = K(\mathbf{X}, \mathbf{X}) - 2K(\mathbf{X}, \mathbf{Y}) + K(\mathbf{Y}, \mathbf{Y})$. The resulting distance function will always correspond to a true distance function, either in Euclidean N-dimensional feature space for some N, or in infinite dimensional space [Berg et al., 1984]. Thus any kernel can be used in an unsupervised pattern recognition method as well as in a supervised pattern recognition method like SVMs, so long as that method relies only on distance (or similarity) calculations, and not on explicit construction of the feature space (see, e.g., [Scholkopf et al., 1999]). This may be a fruitful area for further research.

Finally, we note that a number of researchers have analyzed DNA microarray gene expression data with the goal of reconstructing complete regulatory pathways within the cell. The work we have presented makes no attempt to infer pathways, nor is it obvious how SVM methods could be applied to this much more complex task. However, the functional classification of genes is a prerequisite to reconstructing complete pathways, and so this work does contribute toward this goal. Using only the limited features that were available to us, we have barely scratched the surface of this problem, but the potential is significant.

Acknowledgments

The authors would like to thank Tommi Jaakkola for assistance in the development of the SVM software. Michael P. S. Brown is supported by a PMMB Burroughs Wellcome Predoctoral Fellowship. William N. Grundy is supported by a Sloan/DOE Fellowship in Computational Molecular Biology. David Haussler is supported by DOE grant DE-FG03-95ER62112 and NSF grant DBI-9808007 to David Haussler and NIH grant CA 77813 to David Haussler and Manuel Ares.

A Support vector machines

Support vector machines map a given set of binary labeled training data to a high-dimensional feature space and separate the two classes of data with a maximum margin hyperplane. In general, this hyperplane corresponds to a nonlinear decision boundary in the input space.

Let $\mathbf{X} \in R_0 \subseteq \Re^n$ be the input vectors, $y \in \{-1, +1\}$ be the labels, and $\phi : R_0 \to F$ be the mapping from input space to feature space. Then the SVM learning algorithm finds a hyperplane (w, b) such that the quantity

$$\gamma = \min_i y_i \{\langle w, \phi(\mathbf{X}_i)\rangle - b\} \tag{10}$$

is maximized, where the vector w has the same dimensionality as F, b is a real number, and γ is called the *margin*. The corresponding decision function is then

$$f(\mathbf{X}) = \mathrm{sign}\left(\langle \mathrm{w}, \phi(\mathbf{X})\rangle - \mathrm{b}\right) \tag{11}$$

It is easy to prove [Vapnik, 1998] that this minimum occurs when

$$w = \sum_i \alpha_i y_i \phi(\mathbf{X}_i) \tag{12}$$

where α_i are positive real numbers that maximize

$$\sum_i \alpha_i - \sum_{ij} \alpha_i \alpha_j y_i y_j \langle \phi(\mathbf{X}_i), \phi(\mathbf{X}_j)\rangle \tag{13}$$

subject to

$$\sum \alpha_i y_i = 0, \alpha_i > 0. \tag{14}$$

The decision function can equivalently be expressed as[2]

$$f(\mathbf{X}) = \mathrm{sign}\left(\sum_{\mathrm{i}} \alpha_{\mathrm{i}} \mathrm{y}_{\mathrm{i}} \langle \phi(\mathbf{X}_{\mathrm{i}}), \phi\left(\mathbf{X}\right)\rangle - \mathrm{b}\right). \tag{15}$$

From this equation it is possible to see that the α_i associated with the training point $\mathbf{X}_i$ expresses the strength with which that point is embedded in the final decision function. A remarkable property of this alternative representation is that only a subset of the points will be associated with a non-zero α_i. These points are called *support vectors* and are the points that lie closest to the separating hyperplane. The sparseness of the α vector has several computational and learning theoretic consequences.

It is important to note that neither the learning algorithm nor the decision function (Equation 15) needs to represent explicitly the image of points in the feature space, $\phi(\mathbf{X}_i)$, since both use only the dot products between such images, $\langle \phi(\mathbf{X}_i), \phi(\mathbf{X}_j)\rangle$. Hence, if one were given a function $K(\mathbf{X}, \mathbf{Y}) = \langle \phi(\mathbf{X}), \phi(\mathbf{Y})\rangle$, one could learn and use the maximum margin hyperplane in the feature space without ever explicitly performing the mapping. For each continuous positive definite function $K(\mathbf{X}, \mathbf{Y})$ there exists a mapping ϕ such that $K(\mathbf{X}, \mathbf{Y}) = \langle \phi(\mathbf{X}), \phi(\mathbf{Y})\rangle$ for all $\mathbf{X}, \mathbf{Y} \in R_0$ (Mercer's Theorem). The function $K(\mathbf{X}, \mathbf{Y})$ is called the *kernel function*.

The use of a kernel function allows the support vector machine to operate efficiently in a nonlinear high-dimensional feature spaces without being adversely affected by the dimensionality of that space. Indeed, it is possible to work with feature spaces of infinite dimesion. Moreover, Mercer's theorem makes it possible

[2]An alternate formulation of support vector machines does not use an explicit bias b but makes the bias implicit by adding 1 to the kernel function. In this case, the hyperplane goes through the origin, and the optimization does not require the constraint $\sum \alpha_i y_i = 0$. We use this implicit bias method in our experiments.

to learn in the feature space without even knowing ϕ and F. The matrix $K_{ij} = \langle\phi(\mathbf{X}_i), \phi(\mathbf{X}_j)\rangle$ is called the *kernel matrix* and will be particularly important in the extensions of the algorithm that will be discussed later.

Finally, note that the learning algorithm is a quadratic optimization problem that has only a global optimum. The absence of local minima is a significant difference from standard pattern recognition techniques such as neural networks. For moderate sample sizes, the optimization problem can be solved with simple gradient descent techniques like the ones used in this paper (see also [Campbell and Cristianini, 1999] and [Jaakkola and Haussler, 1998]). For larger problems, more advanced techniques should be used [Platt, 1999].

In the presence of noise, the standard maximum margin algorithm described above can be subject to overfitting, and more sophisticated techniques should be used. This problem arises because the maximum margin algorithm always finds a perfectly consistent hypothesis and does not tolerate training error. Sometimes, however, it is necessary to trade some training accuracy for better predictive power. The need for tolerating training error has led to the development the soft-margin and the margin-distribution classifiers. One of these techniques [Shawe-Taylor and Cristianini, 1999] replaces the kernel matrix in the training phase as follows:

$$K \leftarrow K + \lambda I, \tag{16}$$

while still using the standard kernel function in the decision phase (Equation 15). By tuning λ, one can control the training error, and it is possible to prove that the risk of misclassifying unseen points can be decreased with a suitable choice of λ [Shawe-Taylor and Cristianini, 1999].

If instead of controlling the overall training error one wants to control the trade-off between false positives and false negatives, it is possible to modify K as follows:

$$K \leftarrow K + \lambda D, \tag{17}$$

where D is a diagonal matrix whose entries are either d^+ or d^-, in locations corresponding to positive and negative examples. It is possible to prove that this technique is equivalent to controlling the size of the α_i in a way that depends on the size of the class, introducing a bias for larger α_i in the class with smaller d. This in turn corresponds to an asymmetric margin; i.e., the class with smaller d will be kept further away from the decision boundary [Veropulos et al., 1999].

In this work, the extreme imbalance of the two classes, along with the presence of noise, creates a situation in which points from the minority class can be easily mistaken for mislabelled points. Enforcing a strong bias against training errors in the minority class provides protection agaist such errors and forces the SVM to make the positive examples support vectors. Thus, choosing $d^+ = \frac{1}{n^+}$ and $d^- = \frac{1}{n^-}$ provides a heuristic way to automatically adjust the relative importance of the two classes, based on their respective cardinalities. This technique effectively controls the trade-off between sensitivity and specificity [Veropulos et al., 1999].

References

[Berg et al., 1984] Berg, C., Christensen, J., and Ressel, P. (1984). *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions*. Springer.

[Bishop, 1995] Bishop, C. (1995). *Neural networks for pattern recognition*. Oxford UP.

[Breiman et al., 1984] Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Wadsworth International Group.

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

[Campbell and Cristianini, 1999] Campbell, C. and Cristianini, N. (1999). Simple training algorithms for support vector machines. Submitted for publication, and available at http://lara.enm.bris.ac.uk/cig.

[Chu et al., 1998] 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.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

[DeRisi et al., 1997] 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.

[Dick et al., 1997] Dick, F. A., Karamanou, S., and Trumpower, B. L. (1997). QRS1, an essential yeast gene with a genetic relaionship to a subunit of the mitochondrial cytochrome bc1 complex codes for a 60s ribosomal subunit protein. *J. Biol. Chem.*, 272(20):13372–13379.

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

[Eisen, 1999] Eisen (1999). Cluster analysis and display of genome-wide expression patterns. http://rana.-stanford.edu/clustering.

[Eisen et al., 1998] Eisen, M., Spellman, P., Brown, P., and Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *pnas*, 95:14863–14868.

[Fujimuro et al., 1998] Fujimuro, M., Tanaka, K., Yokosawa, H., and Toh-e, A. (1998). Son1p is a component of the 26S proteasome of the yeast *saccharomyces cerevisiae*. *FEBS Letters*, 423(1):149–154.

[Garrett and Grisham, 1995] Garrett and Grisham (1995). *Biochemistry*. Saunders College Publishing.

[Glickman et al., 1998] Glickman, M. H., Rubin, D. M., Fried, V. A., and Finley, D. (1998). The regulatory particle of the *saccharomyces cerevisiae* proteasome. *Molecular and Cellular Biology*, 18(6):3149–3162.

[Guyon et al., 1996] Guyon, I., Matic, N., and Vapnik, V. (1996). Discovering informative patterns and data cleaning. In Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*, pages 181–203. MIT Press.

[Jaakkola et al., 1998] Jaakkola, T., Diekhans, M., and Haussler, D. (1998). A discriminative framework for detecting remote protein homologies. Unpublished, available from `http://www.cse.ucsc.edu/research/compbio/research.html`.

[Jaakkola et al., 1999] Jaakkola, T., Diekhans, M., and Haussler, D. (1999). Using the Fisher kernel method to detect remote protein homologies. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*.

[Jaakkola and Haussler, 1998] Jaakkola, T. and Haussler, D. (1998). Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems 11*, San Mateo, CA. Morgan Kauffmann Publishers. To appear.

[Johnson et al., 1995] Johnson, E. S., Ma, P. C., Ota, I. M., and Varshavsky, A. (1995). A proteolytic pathway that recognizes ubiquitin as a degradation signal. *J Biol Chem*, 270(29):17442–17456.

[Kinzy and J. L. Woolford, 1995] Kinzy, T. G. and J. L. Woolford, J. (1995). Increased expression of *saccharomyces cerevisiae* translation elongation factor 1 α bypasses the lethality of a TEF5 null allele encoding elongation factor 1 β. *Genetics*, 141(2):481–489.

[Kitajima-Ihara and Yagi, 1998] Kitajima-Ihara, T. and Yagi, T. (1998). Rotenone-insensitive internal NADH-quinone oxidoreductase of *saccharomyces cerevisiae* mitochondria: the enzyme expressed in *escherichia coli* acts as a member of the respiratory chain the host cells. *FEBS Letters*, 421(1):37–40.

[Lashkari et al., 1997] Lashkari, D., DeRisi, J., McCusker, J., Namath, A., Gentile, C., Hwang, S., Brown, P., and Davis, R. (1997). Yeast microarrays for genome wide parallel genetic and gene expression analysis. *pnas*, 94:13057–13062.

[Mannhaupt, 1999] Mannhaupt, G. (1999). Personal communication. Max-Planck-Institut fuer Biochemie.

[Marres et al., 1991] Marres, C. A. M., de Vries, S., and Grivell, L. A. (1991). Isolation and incativation of the nuclear gene encoding the rotenone-insensitive internal NADH:ubiquinone oxidoreductase of mitochondria from *saccharomyces cerevisiae*. *Eur. J. Biochem*, 195(1):857–862.

[MYGD, 1999] MYGD (1999). Munich information center for protein sequences yeast genome database. http://www.mips.biochem.mpg.de/proj/yeast.

[Papa et al., 1999] Papa, F. R., Alexander, Y. A., and Hochstrasser, M. (1999). Interaction of the Doa4 deubiquitinating enzyme with the yeast 26S proteasome. *Molecular Biology of the Cell*, 10(1):741–756.

[Platt, 1999] Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. In Schölkopf, B., Burges, C. J. C., and Smola, A. J., editors, *Advances in Kernel Methods*. MIT Press.

[Quinlan, 1997] Quinlan, J. (1997). C4.5. In *Programs for Machine Learning*, Morgan Kaufmann Series in Machine Learning. Morgan Kaufmann.

[Schölkopf et al., 1997] Schölkopf, B., Sung, K., Burges, C., Girosi, F., Niyogi, P., Poggio, T., and Vapnik, V. (1997). Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE Transactions on Signal Processing*, 45(11):2758–2765.

[Scholkopf et al., 1999] Scholkopf, C., Burges, J., and Smola, A. (1999). *Advances in Kernel Methods: Support Vector Learning*. MIT Press.

[Shawe-Taylor and Cristianini, 1999] Shawe-Taylor, J. and Cristianini, N. (1999). Further results on the margin distribution. In *Proceedings of COLT99*. available at: http://lara.enm.bris.ac.uk/cig/pubs_nf.htm.

[Spellman et al., 1998a] Spellman, P., Sherlock, G., Zhang, M., Iyer, V., Anders, K., Eisen, M., Brown, P., Botstein, D., and Futcher, B. (1998a). Comprehensive identification of cell cycle-regulated genes of the yeast *saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell*, 9:3273–3297.

[Spellman et al., 1998b] Spellman, P. T., Sherlock, G., Zhang, M. Q., Iyer, V. R., Anders, K., Eisen, M. B., Brown, P. O., Botstein, D., and Futcher, B. (1998b). Comprehensive identification of cell cycle-regulated genes of the yeast *saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell*, 9:3273–3297.

[Stoler et al., 1995] Stoler, S., Keith, K. C., Curnick, K. E., and Fitzgerald-Hayes, M. (1995). A mutation in CSE4, an essential gene encoding a novel chromatin-associated protein in yeast, causes chromosome nondisjunction and cell cycle arrest at mitosis. *Genes Dev*, 9(5):573–586.

[Tamayo et al., 1999] 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. *PNAS*, 96:2907–2912.

[Vapnik, 1998] Vapnik, V. (1998). *Statistical Learning Theory*. Wiley.

[Veropulos et al., 1999] Veropulos, K., Campbell, C., and Cristianini, N. (1999). Controlling the sensitivity of support vector machines. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI99)*, Stockholm, Sweden. available at: http://lara.enm.bris.ac.uk/cig/pubs_nf.htm.

[Wool et al., 1995] Wool, I. G., Chan, Y.-L., and Gluck, A. (1995). Structure and evolution of mammalian ribosomal proteins. *Biochem. Cell Biol.*, 73:933–947.

[Wu et al., 1999] Wu, D., Bennett, K., Cristianini, N., and Shawe-Taylor, J. (1999). Large margin decision trees for induction and transduction. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML99)*. available at: http://lara.enm.bris.ac.uk/cig/pubs_nf.htm.