Gene Classification using Expression Profiles: A Feasibility Study*

Michihiro Kuramochi and George Karypis

University of Minnesota
Department of Computer Science & Engineering/
Army HPC Research Center/Digital Technology Center
4-192 EE/CS Building, 200 Union St SE
Minneapolis, MN 55455

{*kuram, karypis*}*@cs.umn.edu*

As various genome sequencing projects have already been completed or are near completion, genome researchers are shifting their focus to *functional* genomics. Functional genomics represents the next phase, that expands the biological investigation to studying the functionality of genes of a single organism as well as studying and correlating the functionality of genes across many different organisms. Recently developed methods for monitoring genome-wide mRNA expression changes hold the promise of allowing us to inexpensively gain insights into the function of unknown genes. In this paper we focus on evaluating the feasibility of using supervised machine learning methods for determining the function of genes based solely on their expression profiles. We experimentally evaluate the performance of traditional classification algorithms such as support vector machines and k-nearest neighbors on the yeast genome, and present new approaches for classification that improve the overall recall with moderate reductions in precision. Our experiments show that the accuracies achieved for different classes varies dramatically. In analyzing these results we show that the achieved accuracy is highly dependent on whether or not the genes of that class were significantly active during the various experimental conditions, suggesting that gene expression profiles can become a viable alternative to sequence similarity searches provided that the genes are observed under a wide range of experimental conditions.

Keywords: gene classification; expression profiles; SVM; k-NN.

1. Introduction

As various genome sequencing projects have been recently completed or are near completion (*e.g.*, microbial, human, *Arabidopsis*), genome researchers are shifting their focus from *structural* genomics to *functional* genomics [14]. Structural genomics represents an initial phase of genome analysis, whose goal is to construct high

*This work was supported in part by NSF ACI-0133464 and ACI-0312828; the Digital Technology Center at the University of Minnesota; and by the Army High Performance Computing Research Center (AHPCRC) under the auspices of the Department of the Army, Army Research Laboratory (ARL) under Cooperative Agreement number DAAD19-01-2-0014. The content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. Access to research and computing facilities was provided by the Digital Technology Center and the Minnesota Supercomputing Institute.

resolution genetic and physical maps as well as complete sequence information of the chromosoms. Functional genomics represents the next phase, that expands the biological investigation to studying the functionality of genes of a single organism as well as studying and correlating the functionality of genes across many different organisms.

Traditionally, researchers have been using sequence data (either nucleotide sequence in the case of genes, or amino acid sequences in the case of proteins) to determine the function of genes and/or the corresponding proteins. This approach relies on the fact that a set of genes that have sufficiently similar sequences also perform the same function. The explosive growth of the amount of sequence information available in public databases has made such an approach particularly accurate and an indispensable tool towards functional genomics. Despite the fact that functional genomic techniques based on sequence homology can provide a wealth of information about the functionality of entire genomes; they also have two inherent limitations. First, in some cases the functional similarity cannot be inferred by sequence information alone as sequence comparisons can be uninformative and even misleading. Second, even though there are many projects for sequencing entire genomes of different species, there will be a lot of species for which we do not and will never have complete sequence information (at least in the next several decades). This is a function of both the cost associated with sequencing as well as the fact that there are a lot of species.

Recently developed methods for monitoring genome-wide mRNA expression changes such as oligonucleotide chips [10], SAGE [33], and cDNA microarrays [30], are especially powerful as they allow us to quickly and inexpensively observe the changes at the differential expression levels of the entire complement of the genome under many different induced conditions. Knowing when and under what conditions a gene or a set of genes is expressed often provides strong clues as to their biological role and function. Already, numerous such experiments involving relatively small genomes are performed at various sites worldwide. In the coming year the number of this type of experiments involving microarrays is expected to increase significantly.

One way of using the data produced by microarray experiments to determine the function of unknown genes is to use clustering algorithms to group together genes that have similar expression profiles. Based on the distribution of known and unknown genes in such clusters, then some information about the function of previously unknown genes can be inferred. In fact a large number of studies have already taken place in which putative functions of unknown genes have been identified in this way. However, clustering being an unsupervised learning method is not ideally suited for this particular task as it has no mechanism by which to perform feature selection. A better approach of inferring the function of unknown genes based on their expression profiles is to use machine learning techniques based on supervised learning [24].

This has been recently recognized by a number of researchers and a few attempts have been made to use such algorithms. In particular, Golub *et al.* [13], by looking

at expression profiles of a subset of human genes, a particular type of leukemia can be distinguished from another type of the disease. Brown *et al.* [3,2] used several classification algorithms to predict if a gene has a particular function based on expression profiles and obtain encouraging results. Hvidsten *et al.* [15] applied rule-based induction to predict human gene functionality based on the gene ontology database [11] from expression profiles of the fibroblast serum response [16] and showed high prediction accuracy for 16 gene functional classes. Nevertheless, most of these studies were limited as they focused on only a small set of specific functions and/or did not provide any insights on the overall feasibility of this type of approach for determining the function of the genes.

The focus of this paper is to perform a study on the suitability of supervised learning techniques for determining the function of genes using solely gene expression data and attempts to identify the requirements under which such an approach will lead to accurate predictions. Our work focuses on the yeast genome and uses publicly available microarray datasets [9,8] and covers a large number of gene functions defined in the Munich Information Centre for Protein Sequences (MIPS) database [20,19,21,22]. We present a detailed experimental study using two popular classification algorithms, support vector machines and k-nearest neighbors for predicting the functions of the genes, and present fixed-size prediction algorithms that allow us to trade recall for precision. Our experimental results show that the accuracy achieved by the proposed approaches varies widely depending on the function that we try to predict. For certain classes we can achieve high accuracies and for some classes the accuracies are quite poor. Our analysis shows that the accuracy achieved for a particular class is highly dependent on whether or not the genes of that class were significantly active during the various experimental conditions. This suggests that gene expression profiles can become a viable alternative to sequence similarity searches provided that the genes are observed under a wide range of experimental conditions that *exercise* the various cellular functions.

The rest of this paper is organized as follows. Section 2 describes the source and the structure of two datasets we use in our study, expression profiles and gene functional class assignment. Section 3 explains the detail of binary classification algorithms, support vector machines and the k-nearest neighbors. We will also propose two different types of fixed-size prediction algorithms. The results and the evaluation of the experiments are shown in Section 4 and we discuss the relationship between those prediction accuracy results and statistical measure of expression profiles in Section 5. Finally, Section 6 provides some concluding remarks.

2. Datasets Description

As discussed in the introduction our goal is to develop algorithms for determining the function of the yeast genes using supervised learning methods that are based entirely on gene expression data. In order to achieve that we need to have access to two key pieces of information: (i) the actual expression profiles, and (ii) the different

functional classes that the various genes belong to. These are described in the rest of this section.

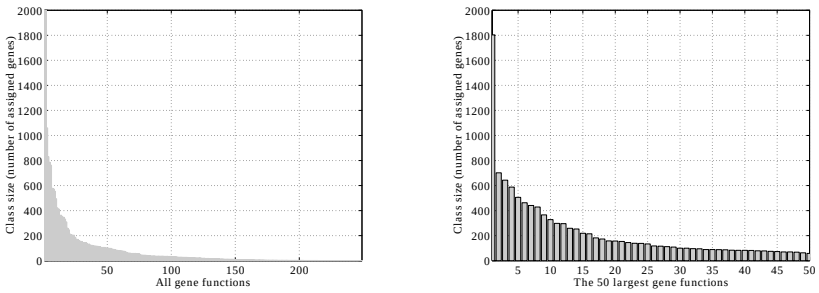
2.1. *Expression Profiles*

In our study we used the publicly available expression profiles from Brown's group at Stanford University [8,9]. The source of these profiles were 8 different microarray experiments under different conditions. They can be categorized into the following 4 types, (i) the mitotic cell division cycle, (ii) sporulation, (iii) temperature and reducing shocks, (iv) gene expression in the the budding yeast during the diauxic shift. These experiments resulted in a total of 79 measurements, however, not all genes have the entire set of the 79 measurements because each experiment was performed on a different subset of genes. We treat those missing values as zero. The 79 measures are base 2 logarithms of ratios of intensities scanned from two separate fluorescence dye images, which were obtained after hybridization. Even though the whole yeast genome contains 6275 genes, the arrays used in the above experiments contained only 2467 genes. Out of expression profiles for those 2467 genes, we used 2462 profiles by discarding profiles for genes that do not appear in the MIPS database. Section 2.2 describes the database in detail.

2.2. *Gene Functional Class Assignment*

Determining the functional class of the different genes is very much an ongoing process and to a large extent one of the key steps in understanding the genomes of the various species. Fortunately, in the case of yeast, there exist extensive annotations for a large fraction of the genes. For our study we used the functional annotations that are available in the MIPS database [20,19,21,22]. As of the time of this writing, the MIPS database defines a total of 249 gene functional classes, organized in a tree structure.

Based on the amount of information that is known for each gene, the MIPS database assigns it to one or more nodes of the tree of function classes. Genes for which detailed functional information is known tend to be assigned towards the leaves of the tree (*i.e.*, more specific classes), whereas genes for which the information is more limited tend to be assigned at the higher-level nodes of the tree, (*i.e.*, more abstract classes). Out of the total number of 6275 genes of the yeast genome, MIPS provides at least one annotation for 3902 genes. For example, a gene YBR069C is assigned a function named *amino-acid transport*. Because *amino-acid transport* is a sub-function of *amino-acid metabolism* which is also a sub-function of the top-level function *METABOLISM*, YBR069C has all those functions, {*amino-acid transport*, *amino-acid metabolism*, *METABOLISM*}, *i.e.*, a function at a node and all the functions of its path to the top-level node. A gene also may have functions assigned from multiple branches. For the case of YBR069C, it has functions from the top level category *METABOLISM* and its subcategories as well as ones from other top level classes *TRANSPORT FACILITATION*, *CELLULAR TRANSPORT AND*



(a) All gene functions (b) The 50 largest functions

Fig. 1. Distribution of the size of gene functions

TRANSPORTMECHANISMS and *CELLULAR ORGANIZATION* and their subcategories. As a result of this functional class assignment, each gene has 3.4 functions assigned on the average. All the 2462 genes in the expression profile dataset described in Section 2.1 do have at least one functional annotation. The distribution of the number of classes at the different levels of the tree is shown in Table 1.

Table 1. Number of defined function categories at each level in the tree structure

Level	Functions
1	16
2	107
3	85
4	39
5	2

Figure 1(a) shows the size of the different gene functions in the MIPS class assignment. By "size", we mean the number of genes assigned to the corresponding function. Most of the functions are small in their size, which makes functionality prediction via supervised machine learning approaches difficult. For this reason, we focus only on the 50 largest functional classes whose size distribution is shown in Figure 1(b). The name of those function categories are shown in Table 2. Parenthesized numbers in each function class show the number of genes assigned to the category in the profile dataset, and indentation corresponds to the depth of each function category. On the average, a gene has 4.6 gene functions.

3. Methods

The goal of supervised learning methods, also known as classification algorithms, is to build a set of models that can correctly predict the class of the different ob-

Table 2. The 50 largest functions in the expression profile dataset

(1) *METABOLISM* (1061)
  (a) *amino-acidmetabolism* (204)
    i. *amino-acid biosynthesis* (118)
  (b) *nucleotide metabolism* (144)
  (c) *C-compound and carbohydrate metabolism* (414)
    i. *C-compound and carbohydrate utilization* (261)
    ii. *regulation of C-compound and carbohydrate utilization* (120)
  (d) *lipid, fatty-acid and isoprenoid metabolism* (213)
    i. *lipid, fatty-acid and isoprenoid biosynthesis* (118)
(2) *ENERGY* (247)
(3) *CELL GROWTH, CELL DIVISION AND DNA SYNTHESIS* (832)
  (a) *budding, cell polarity and filament formation* (172)
  (b) *pheromone response, mating-type determination, sex-specific proteins* (161)
  (c) *DNA synthesis and replication* (91)
  (d) *recombination and DNA repair* (99)
  (e) *cell cycle control and mitosis* (347)
(4) *TRANSCRIPTION* (787)
  (a) *rRNA transcription* (106)
  (b) *tRNA transcription* (83)
  (c) *mRNA transcription* (575)
    i. *mRNA synthesis* (422)
      A. *transcriptional control* (333)
    ii. *mRNA processing (splicing)* (106)
(5) *PROTEIN SYNTHESIS* (351)
  (a) *ribosomal proteins* (208)
(6) *PROTEIN DESTINATION* (579)
  (a) *protein targeting, sorting and translocation* (139)
  (b) *protein modification* (187)
  (c) *assembly of protein complexes* (93)
  (d) *proteolysis* (154)
    i. *cytoplasmic and nuclear degradation* (98)
(7) *TRANSPORT FACILITATION* (310)
(8) *CELLULAR TRANSPORT AND TRANSPORTMECHANISMS* (495)
  (a) *vesicular transport (Golgi network, etc.)* (125)
  (b) *cellular import* (101)
(9) *CELLULAR BIOGENESIS* (205)
  (a) *biogenesis of cell wall (cell envelope)* (107)
(10) *CELL RESCUE, DEFENSE, CELL DEATH AND AGEING* (363)
  (a) *stress response* (170)
  (b) *DNA repair* (88)
(11) *IONIC HOMEOSTASIS* (123)
  (a) *homeostasis of cations* (113)
(12) *CELLULAR ORGANIZATION* (2254)
  (a) *organization of plasma membrane* (144)
  (b) *organization of cytoplasm* (556)
  (c) *organization of cytoskeleton* (106)
  (d) *organization of endoplasmatic reticulum* (155)
  (e) *organization of Golgi* (79)
  (f) *nuclear organization* (764)
  (g) *mitochondrial organization* (364)

jects. The input to these methods is a set of objects (training set), the classes that these objects belong to (dependent variable), and a set of variables describing dif-

ferent characteristics of the objects (independent variables). Once such a predictive model is built, then it can be used to predict the class of the objects for which class information is not known *a priori*. For the problem of classifying genes based on their expression profiles, the independent variables are the 79 gene expression levels obtained during the eight different experiments, and the dependent variable is the function of the gene. The key advantages of supervised learning methods over unsupervised methods such as clustering, is that by having an explicit knowledge of the classes the different objects belong to, these algorithms can perform an effective feature selection (*e.g.*, ignoring some of the independent variables) if that leads to better prediction accuracy.

Over the years a variety of different classification algorithms have been developed by the machine learning community. Examples of such algorithms are decision tree based [1,26,25], rule-based [4,5], probabilistic [18], neural networks [23,34], genetic [12], instance-based [7,35], and support vector machines [31,32]. Depending on the characteristics of the data sets being classified certain algorithms tend to perform better than others. In recent years, algorithms based on the support vector machines and the k-nearest neighbors have been shown to produce reasonably good results for problems in which the independent variables are continuous and homogeneous (*e.g.*, they measure a similar quantity). For this reason, our study uses primarily these two classification algorithms.

Support Vector Machines Support vector machines (SVM) is a relatively new learning algorithm proposed by Vapnik [31,32]. This algorithm is introduced to solve two-class pattern recognition problems using the Structural Risk Minimization principle [31,6]. Given a training set in a vector space, this method finds the *best* decision hyperplane that separates two classes. The quality of a decision hyperplane is determined by the distance (referred as margin) between two hyperplanes that are parallel to the decision hyperplane and touch the closest data points of each class. The *best* decision hyperplane is the one with the maximum margin. By defining the hyperplane in this fashion, SVM is able to generalize to unseen instances quite effectively. The SVM problem can be solved using quadratic programming techniques [31,6]. SVM extends its applicability on the linearly non-separable data sets by either using soft margin hyperplanes, or by mapping the original data vectors into a higher dimensional space in which the data points are linearly separable. The mapping to higher dimensional spaces is done using appropriate kernel functions, resulting in efficient algorithms. A new test object is classified by looking on which side of the separating hyperplane it falls and how far away it is from it.

***k*-Nearest Neighbors** k-nearest neighbors (kNN) is a well-known and widely used instance-based classification algorithm. The basic idea behind this classification paradigm is to compute the similarity between a test object and all the objects in the training set, select the k most similar training set objects, and determine the class of the test object based on the classes of these k nearest neighbors. One of the

advantages of kNN is that it is well suited for multi-modal classes as its classification decision is based on a small neighborhood of similar objects. As a result, even if the target class is multi-modal (*i.e.*, consists of objects whose independent variables have different characteristics for different subsets), it can still lead to good classification accuracy.

Two steps are critical to the performance of the kNN classification algorithm. The first is the method used to compute the similarity between the test object and the objects in the training set, and the second is the method used to determine the class of the test object based on the classes of the nearest neighbors. For data sets in which the objects are represented by multi-dimensional vectors, like the gene expression data used in this study, two approaches are commonly used to compute the similarity. The first approach is based on using a Euclidean distance (or any other norm-based distance) between the test object and the training objects, whereas the second approach is based on using the cosine of the angle between the two vectors. The primary difference between these two distance measures, is that the Euclidean distance approach is affected by the length of the test objects whereas the cosine-based approach is length invariant and only focuses in the angles of the two vectors. Recent studies using gene expression data [3] have shown that cosine-based similarity functions are better as they focus on the relative shape of the profile and not its magnitude. For this reason, in our experiments the similarity between two genes was computed using the cosine function which is defined as follows. If $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are the two vectors, then their cosine similarity is given by

$$\cos(\boldsymbol{v}_i, \boldsymbol{v}_j) = \boldsymbol{v}_i \cdot \boldsymbol{v}_j / \|\boldsymbol{v}_i\| \ \|\boldsymbol{v}_j\|,$$

where "$\cdot$" denotes the dot-product between two vectors, and $\|\boldsymbol{v}\|$ denotes the 2-norm (*i.e.*, length) of the vector.

The simplest way to determine the class of the test object based on the classes of its k-nearest neighbors is to assign it to the majority class, *i.e.*, the class in which most of the k-nearest objects belong to. This approach can be easily extended to weighting differently the different neighbors based on the actual similarity. In this case, instead of simply adding the frequencies of the individual classes we do so in a weighted fashion based on how similar a particular neighbor is to the test object. If the training set contains only two classes, the *positive* and *negative* class, then this can be done by looking at the value of the measure q that is defined as:

$$q = \sum_{i=1}^{k} \cos(\boldsymbol{v}_i, \boldsymbol{v}) c(\boldsymbol{v}_i), \tag{1}$$

where

$c(\boldsymbol{v}_i) = \{\, 1$ if $\boldsymbol{v}_i$ belongs to the positive class, -1if $\boldsymbol{v}_i$ belongs to the negative class.

If q is positive, then it is assigned to the positive class, otherwise it is assigned to the negative class.

3.1. *Binary Classification*

Traditional classification algorithms are primarily suited for learning classification models in which each object belongs to only a single class. Nevertheless, in our data set each gene has more than one classes associated with it. A common way of solving this type of classification problems is to build a set of binary classifiers, each distinguishing the genes of one functional class from the genes that do not belong to this class. We will refer to the particular functional class as the *positive* class, and the rest of the genes as the *negative* class. For our problem this leads to 50 different binary classifiers, one for each gene function. Once the classifiers have been built, a new gene is classified by testing it against each one of the 50 binary classifiers. Each gene is then assigned to all the classes for which the particular classifier determined that it was part of the positive class.

Given a set of genes for which we already know their classes and where not used during training we can use a particular binary classifier to predict their classes. By comparing how many of them are predicted to be in the positive class we can then evaluate its predictive performance. By combining the predictions with the actual classes we can partition the test genes into four classes. The *true positives* and the *true negatives* which are the set of genes that were correctly predicted to be part of the positive or negative class, respectively; and the *false positives* and *false negatives* which are the sets of genes that were incorrectly predicted as positives or negatives, respectively. A common way of measuring that performance is to use two measures called the *precision* and *recall*. The precision p of a binary classifier is defined as

$$p = \frac{N_{\text{true positives}}}{N_{\text{true positives}} + N_{\text{false positives}}},$$

and the recall is defined as

$$r = \frac{N_{\text{true positives}}}{N_{\text{true positives}} + N_{\text{false negatives}}}.$$

The precision measures what fraction of the genes that are predicted positive are actually positive, and the recall measures what fraction of the positive genes were actually predicted as positive. An alternate way of evaluating the performance of a classifier is to look at its accuracy, which is defined as the fraction of correct predictions. However, when the different classes are of significantly different sizes, the accuracy measure can be misleading, and looking at precision and recall provides more meaningful information.

In the SVM algorithm the classification decision is made by looking at how far a test object is from the decision hyperplane, whereas in the case of the kNN algorithm, the classification is made by looking at the q measure defined in equation (1). If the distance to the hyperplane or the value of q is positive the algorithms assign an object to the positive class. Essentially, in both of these algorithms the value zero acts as a threshold in determining the class of the object. However, in many cases a threshold value that is greater or smaller than zero may be more appropriate. To avoid the arbitrariness of this particular threshold setting, we set a threshold for

classifiers at a value called the *break-even point* where the precision and the recall becomes equal. In general, if the value of the decision threshold increases (*i.e.*, it becomes harder to assign something to the positive class) the precision increases and the recall decreases. On the other hand, if the decision threshold decreases the precision will tend to decrease and the recall will tend to increase. By changing the value of the decision threshold we can then find the point at which the precision becomes the same as the recall.

3.1.1. *Fixed-size Predictions*

As discussed in the previous section the approach based on binary classifiers can be used to address the problem of classifying genes into multiple classes. Nevertheless one limitation of that approach is that it does not allow us to directly control the number of classes that each gene is assigned to. In some cases we may want to determine for each gene a set of m classes that it will most likely belong to. This is particularly important if expression profile based gene classification is used to identify a set of genes that we may want to study further, for example to obtain their sequences.

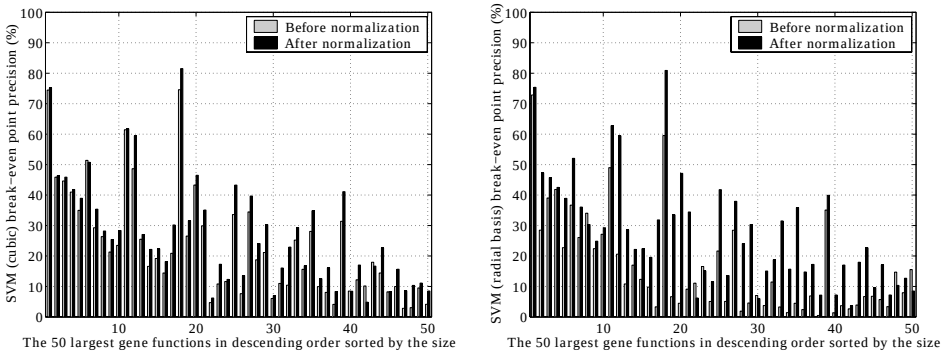
In this study we explored two different approaches for determining the m most likely functions of a gene. The first approach is based on obtaining the list of candidate functions by utilizing the results of the 50 binary classifiers, whereas the second approach is based on finding these candidate classes directly.

As discussed in Section 3.1, for each of the binary classes, both the SVM and the kNN classifiers compute a quantity that essentially measures how strong a particular genes belongs to a particular class. Our first approach for identifying the m most likely functions is based on using these strength measures of the different binary classifiers. In the case of SVM, for a gene we compute its distance to the 50 decision hyperplanes, and assign it to classes that correspond to the m largest values (*i.e.*, strongest predictions). Similarly, in the case of kNN we compute the q measure for each of the 50 classifiers and assign it to the classes that correspond to the m largest values. We will refer to these two approaches as the *SVM-induced* and the *kNN-induced* methods, respectively.

Our direct approach for determining the m most likely candidate functions is based on the kNN approach. In particular, for each gene g_i we identify a set of its k most similar genes, N_{g_i}. We then compute the similarity-weighted frequency of the various classes that the genes in N_{g_i} belong to, and select the m most frequent classes as the predicted classes. This approach was motivated by similar algorithms developed by the information retrieval community for building recommender agents [28,27,29]. We will refer to this approach as the *direct kNN* method.

4. Experimental Results

As discussed in Section 2, because some of the 249 gene function classes defined in the MIPS database cover a very small number of genes, our experimental evaluation



(a) Cubic kernel (b) Radial basis kernel

Fig. 2. Kernel types of SVM and the effect of normalization

was focused only on the 50 largest classes shown in Table 2, using the classifications and datasets described in Section 2. In the rest of this section we present the results for binary classification and fixed-sized classification.

4.1. *Binary Classification Results*

We applied the SVM and the kNN algorithms to predict gene functionality of a subset of the yeast genome. To evaluate prediction accuracy of each algorithm, we performed 3-way cross validation. Each prediction measure is obtained at the break-even point where its recall and precision are equal. The implementation of SVM we used is SVM^{light}, version 3.50 by Joachims [17]. Among various types of kernels that SVM^{light} supports, we chose linear, polynomial (quadratic and cubic) and radial basis functions. We also specified a trade-off option "-c 100" to finish the learning program in reasonable running time. Other parameters are all used as their default settings.

In the case of SVM classification algorithm two parameters were found to play an important role in the overall quality of the results. The first is how the different 79-dimensional vectors representing each gene are normalized and the second is the choice of the kernel. To evaluate the sensitivity on the vector normalization, we performed two sets of experiments. In the first set, we used raw log-ratios of expression levels whereas in the second experiment expression levels were normalized so that each vector is of unit length. Figures 2(a) and 2(b) show the precision of the break-even point achieved by the two representations for the cubic and the radial basis functions. Note that the classes are displayed in decreasing class size order. Looking at these graphs, we can see that in general, normalized representations lead to dramatic improvements for some classes especially with the radial basis kernel.

Next we compare the average precision at the break-even point for different types

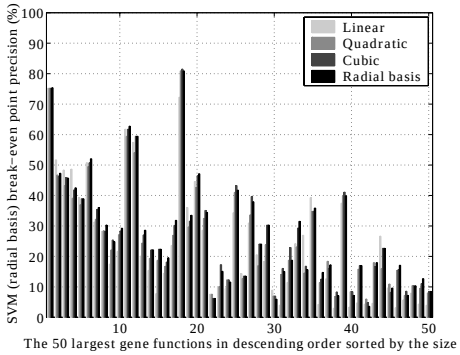


Fig. 3. Comparison of 4 different SVM kernel types

of the SVM kernels (see Figure 3). The average precision over the 50 functions for the linear, the quadratic, the cubic and the radial basis kernels are 23.8%, 25.6%, 27.5% and 27.6% respectively, and the difference by the kernel type is in general small. The performance of the cubic function is similar to that of the radial basis function and those two types outperform the linear and the quadratic kernels. With 27 out of the 50 functions, the radial basis kernel's precision is better than the cubic kernel's. Both the cubic and the radial basis kernels outperform linear and quadratic for more than 31 functions, respectively.

In the case of kNN classifier, we performed a sequence of experiments in which we set the number of neighbors, k, to be 1, 2, 5, 10, 20, 30 and 40. The average precision at the break-even point achieved in this sequence of experiments was 24.4%, 24.7%, 26.2%, 26.4%, 25.9%, 24.9% and 24.2% respectively. Thus, with 10 neighbors kNN shows the best results, however, the number of neighbors has less impact on the break-even point precision compared to the type of the SVM kernels.

Finally, Figure 4 shows the binary classification results of SVM and kNN for the 50 functions. For the SVM classifier we used normalized gene vectors with the radial basis kernel, which achieved the better result than the other polynomial kernels. For the kNN classifiers we used 10 neighbors, because again using 10 neighbors achieved the best precision at the break-even point on the average. From these results we can see that in general SVM achieves slightly better precision than kNN does. Nevertheless, only a few classes can be identified with reasonably high precision, regardless of the classification methods. The average precision for both radial basis SVM and kNN with $k = 10$ was 27.6% and 26.7%, respectively. Note that there is tendency that larger functions are easy to get correct prediction than smaller ones.

4.2. *Fixed-size Prediction Results*

Motivated by the relatively poor results obtained by the binary classification algorithms, we focused on developing algorithms that for each gene, predict a fixed number m of candidate classes. The key goal of this approach is to try to achieve

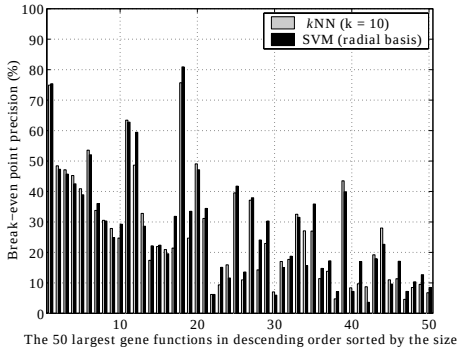


Fig. 4. Precision at the break-even point of predicting gene functionality with SVM and kNN

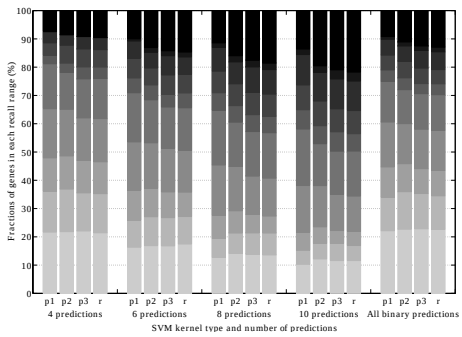
a higher level of recall—*i.e.*, to predict most of the classes of a particular gene—at the cost of potentially achieving a somewhat lower precision.

As discussed in Section 3.1.1, we developed three schemes: the SVM-induced and the kNN-induced schemes that obtain predictions using the 50 individual binary classifiers, and the direct kNN scheme that uses the kNN-type algorithm to directly compute these predictions.

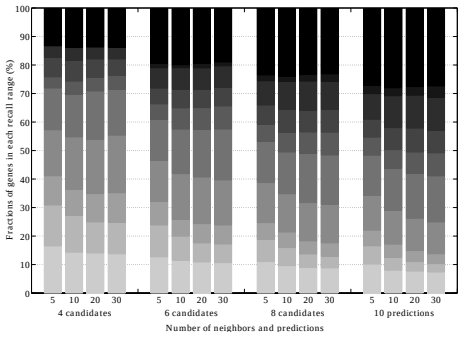
Figure 5 shows the results obtained in this set of experiments with all the three schemes, under different values of m, and under different parameters of the underlying classification algorithms. In the case of the SVM-induced method, Figure 5(a) shows the results obtained using the linear ("p1"), the quadratic ("p2"), the cubic ("p3") and the radial basis ("r") kernels. In the case of the kNN-induced method, Figure 5(c) shows the results achieved for k equal to 5, 10, 20 and 30. In the case of the direct kNN method, Figure 5(b) shows the results achieved by using a neighborhood of size 5, 10, 20 and 30. Also for comparison purposes, the results labelled "All binary predictions" in Figures 5(a) and 5(c) show the results obtained by only using the binary predictions at the break-even point. Note that unlike the fixed-size results, in these two sets of results, the number of predictions made for each gene is not uniform.

Looking at the different fixed-size prediction results we can see that as expected the overall recall increases as we increase the number of predictions m. Nevertheless, as m increases, the overall precision decreases. Comparing the results produced by the SVM-induced method with those produced by the kNN-induced method, we can see they are quite similar (at least for the radial basis kernel and the 10 nearest neighbors).

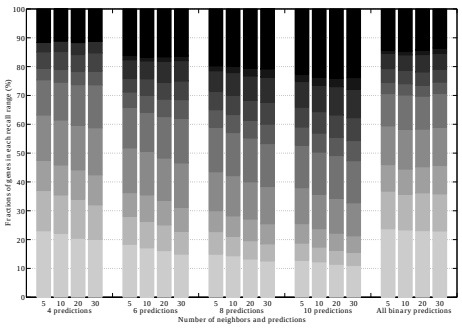
On the other hand, the direct kNN method outperforms the other two and always produces at least 4% higher precision and 5% higher recall at each corresponding experiment. This can be easier seen by looking at the results in Table 3 that summarizes the overall precision and recall of the various schemes for their best set of parameters. The overall results indicate that relatively high levels of recall

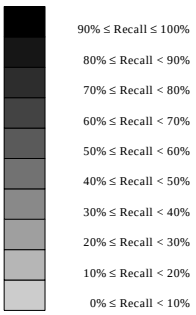


(a) Fixed-size prediction with the SVM-induced classifiers



(b) Fixed-size prediction with the direct kNN method





(c) Fixed-size prediction with the kNN-induced classifiers

Fig. 5. Fractions of genes and their recall in the fixed-size prediction scheme. For the SVM-induced classifier, we used the linear ("p1"), the quadratic ("p2"), the cubic ("p3") and the radial basis ("r") functions as their kernel. For both the kNN-induced classifier and the direct kNN classifier, 5, 10, 20 and 30 neighbors are used.

can be obtained with a moderate reduction in precision. For instance, for $m = 6$, 2.4 ($\simeq 6 \times 0.398$) predictions out of 6 from the direct kNN classifier are likely to be correct, and by those predictions we can discover all functions of every incoming gene with 51.6% probability.

5. Analysis of Results

The experimental results presented in Section 4 showed that the precision of the predictions produced by either the SVM or the kNN classification algorithms varies widely for different functional classes. For some classes we were able to achieve high precision at the break-even point whereas for some of the other classes the

Table 3. Average precision and recall of 3 fixed-size prediction schemes with 4, 6, 8 and 10 predictions. For the SVM-induced classifier and the kNN-induced classifier, the rows with the prediction "all" show the results by considering all the prediction returned by each of the 50 induced classifiers, without limiting the number of predictions.

Method	Predictions	Precision[%]	Recall[%]
SVM-induced (radial basis)	4	41.9	37.0
	6	34.5	44.9
	8	29.8	51.3
	10	26.1	55.9
	all	41.8	40.3
kNN-induced (10 neighbors)	4	42.5	37.7
	6	35.2	45.8
	8	29.8	51.2
	10	26.0	55.8
	all	43.5	40.4
Direct kNN (10 neighbors)	4	48.6	43.4
	6	39.8	51.6
	8	33.7	56.8
	10	30.0	61.3

precision was extremely low. In this section we attempt to analyze these results and understand both the limitations and advantages of the proposed approach for gene classification.

Our analysis will primarily focus on relating the classification accuracies with some of the properties of the gene expression data sets. In particular we will focus on the following characteristics: (i) class size, (ii) class homogeneity, (iii) variability of the expression profiles, and (iv) the level of the differentially expressed profiles.

As discussed in Section 2.2, the number of genes contained in the 50 largest functional classes that were used in our dataset varied significantly. To see if there is a relation between the size of the class and the prediction quality we plotted the size versus the precision at the break-even point for all the 50 classes achieved by the SVM classifier. These results are shown in Figure 6. From this plot we can see that, in general, if the size of the class is large the precision that was obtained is quite high. Nevertheless, the opposite is not true, as for some small classes, SVM was able to achieve precisions that are quite high. Also, the fact the larger classes achieve better precision at the break-even point should not be surprising, as they are easier to classify even by a random classifier.

The second characteristic that we focused was whether or not the "tightness" of a particular class played some role in determining the overall quality of the predictions. In determining how tight a particular class is we computed the average pairwise similarity between the genes in each class using the cosine similarity function. Figures 7(a) and 7(b) plot the size of each class versus its tightness and the precision achieved for each class versus its tightness. From Figure 7(a) we can see that there is relatively little variation in the tightness of the different classes, with the exception of a few classes that are particularly tight. From Figure 7(b) we can see that class tightness does not play a significant role in determining the precision of the classifier. For a relatively narrow range of class tightness values, the accura-

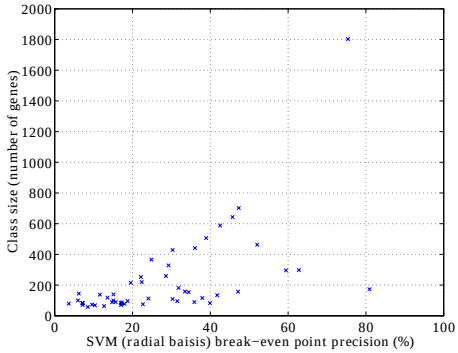
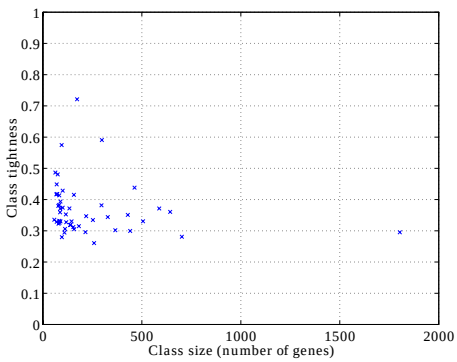
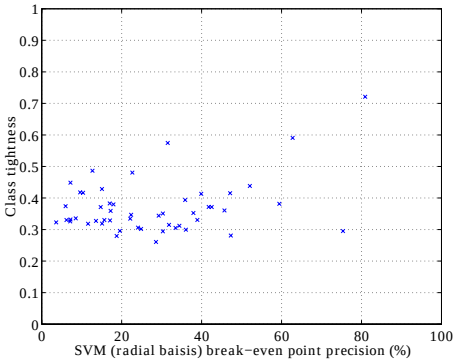


Fig. 6. SVM (radial basis function) binary classification precision at the break-even point and class size



(a) Class size and tightness

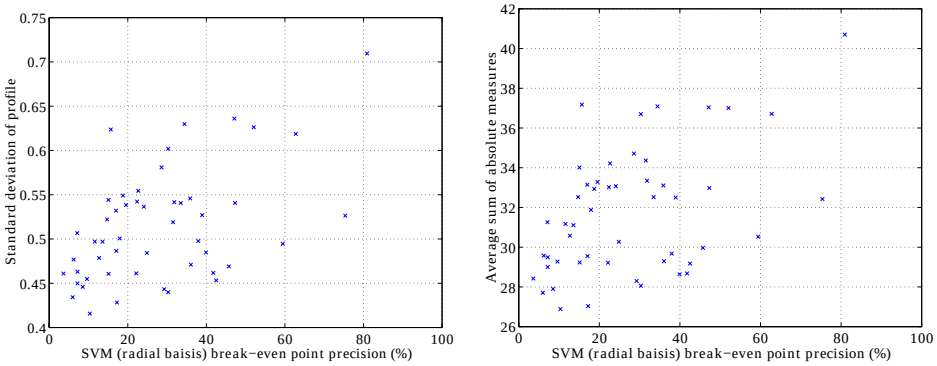


(b) Break-even point precision and class tightness

Fig. 7. Tightness and class size of the 50 largest functional categories

cies obtained differ dramatically and there are classes that achieve high precision which are not tight and vice versa.

The last set of characteristics that we focused has to do with whether or not the variability or the strength of the differential expression of the genes in a class was critical for achieving high precisions. Figures 8(a) and 8(b) plot the precision versus the average standard deviation of the expression profiles of each class as the index of the signal variability, and the average sum of the absolute expression levels as the index of the signal strength, respectively. We computed the average standard deviation by computing the standard deviation of an expression profile over all the 79 measurements of each gene that belongs to a particular class and taking the average of those standard deviations of genes in the class. Similarly, the average absolute sum was computed by adding absolute values of all the 79 measurements



(a) Break-even point precision and standard deviation of profiles

(b) Break-even point precision and average sum of absolute values in profiles

Fig. 8. SVM binary classification precision at the break-even point and function class statistical properties

of each gene in a particular class, and taking the average of those summations in a class. In computing both the standard deviation and the sum of the expression levels we used the log-ratios of the intensities of target versus control, without the unit-length normalization that we used for the classification experiments. Looking at these results we can see that there is a relation between the precisions achieved by SVM and the variability of the profiles or their overall differential expression level. The higher the variability of overall expression levels the higher the precision that was obtained. We computed the correlation coefficient for the two plots and we found that they are 0.51 for Figure 8(a) and 0.47 for Figure 8(b).

Table 4. Classes with high precision at the break-even point

Function Class	P	S	T	σ	Σ
ribosomal proteins	80.9	173	0.721	0.710	40.7
CELLULAR ORGANIZATION	75.4	1803	0.295	0.526	32.4
PROTEIN SYNTHESIS	62.8	298	0.591	0.619	36.7
mitochondrial organization	59.4	296	0.382	0.495	30.5
organization of cytoplasm	52.1	463	0.438	0.626	37.0
METABOLISM	47.3	702	0.281	0.541	33.0
ENERGY	47.1	157	0.415	0.636	37.0
nuclear organization	45.7	643	0.361	0.469	30.0

P: SVM precision[%]; S: size; T: tightness; σ: standard deviation; Σ: absolute sum.

The correlation between the variability or the absolute levels of expression change and whether or not we can accurately predict them should not be surprising as in these type of classes the genes tend to exhibit a distinctly different

behavior that can be used by the classification algorithms to build accurate models for predicting them. On the other hand, if a class contains genes that either have not been *turned on* during the experiments or they have a relative constant profile, the classification algorithms cannot reliably distinguish them from genes of similarly behaving classes. Our analysis indicates that in order for the genes of a particular class to be predicted accurately, the microarray experiments that are performed must have either activated or surpassed them. Unfortunately, the eight different microarray experiments used in deriving our data set were primarily focuses on a small set of cellular functions so do not provide a sufficient breadth. We believe, however, that as additional and more diverse experiments are performed, supervised learning is a viable method for determining the functions(s) of a gene. To further illustrate this point, Table 4 shows the eight classes that achieved the highest precision along with the values of their different characteristics. From the description of the experiments that used in obtaining the microarray data (Section 2.1) and the studies reported in [9,2,3] the genes in these functional classes were shown to be active in the course of the experiments.

6. Conclusions

In this paper we explored the possibility of using microarray expression profiles to find the functions of genes. We applied two representative binary classification algorithms, SVM and kNN for the 2462 annotated genes out of all the 6275 identified genes from the yeast genome. The goal of the classification was to predict functional categories of genes defined in the MIPS database. Because of nonuniform distribution of genes over functional classes, we focused on the 50 largest gene functional categories out of 249. The results showed that the overall prediction accuracy was poor except a few functional categories which are closely related with the nature of the experimental conditions for obtaining expression profiles.

Provided that the binary classifiers produced the low prediction precision, using SVM and kNN as the underlying modules we developed three different schemes, the SVM-induced, the kNN-induced and the direct kNN methods that can predict a specified number of candidate functions in order to achieve high recall, and evaluated their prediction performance in terms of precision and recall. It turned out that the direct kNN approach outperforms the other two. Compared with the binary classification results, those three fixed-size prediction schemes improved recall without a significant loss of precision.

To understand those results of the binary and the fixed-sized prediction approaches, we analyzed the relationship between the binary prediction precision of each function class and statistical measures of the DNA expression profiles, which revealed that (i) large functional classes are relatively easy to predict correctly, and (ii) the variability of expression profiles has influence on the prediction precision.

Based on those experiments and the analysis we believe it will be feasible to make use of expression profiles collected under appropriate conditions to predict

gene functionality as more diverse experiments are performed and pre-examined data are accumulated.

References

1. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman & Hall, 1984.
2. M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. W. Sugnet, M. Ares, Jr., and D. Haussler. Support vector machine classification of microarray gene expression data. Technical Report UCSC-CRL-99-09, Department of Computer Science, University of California, Santa Cruz, 1999.
3. M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. W. Sugnet, T. S. Furey, M. Ares, Jr., and D. Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. In *Proc. Natl. Acad. Sci.*, volume 97, pages 262–267, 2000.
4. R. M. Cameron-Jones and J. R. Quinlan. Efficient top-down induction of logic programs. *SIGART Bulletin*, 5(1):33–42, 1994.
5. W. W. Cohen. Fast effective rule induction. In *Proc. the 12 Int. Conf. on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.
6. C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.
7. B. V. Dasarathy. *Nearest neighbor (NN) norms: NN pattern classification techniques*. IEEE Computer Society Press, 1991.
8. M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Bostein. Cluster analysis and display of genome-wide expression patterns, on-line supplement. http://genome-www.stanford.edu/clustering/Figure2.txt.
9. M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Bostein. Cluster analysis and display of genome-wide expression patterns. In *Proc. of Natl. Acad. Sci.*, volume 95, pages 14863–14868, 1998.
10. S. P. Fodor, R. P. Rava, X. C. Huang, A. C. Pease, C. P. Holmes, and C. L. Adams. Multiplexed biochemical assays with biological chips. *Nature*, 364:555–556, 1993.
11. The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genet.*, 25:25–29, 2000.
12. D. E. Goldberg. *Genetic Algorithms in Search, Optimizations and Machine Learning*. Addison-Wesley, Reading, MA, 1989.
13. T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.
14. P. Hieter and M. Boguski. Functional genomics: It's all how you read it. *Science*, 278:601–602, 1997.
15. T. R. Hvidsten, J. Komorowski, A. K. Sandvik, and A. Laegreid. Predicting gene function from gene expressions and ontologies. In *Proc. of The Pacific Symposium on Biocomputing*, 2001.
16. V. R. Iyer and M. B. Eisen. The transcriptional program in the response of human fibroblasts to serum. *Science*, 283:83–87, 1999.
17. T. Joachims. *Advances in Kernel Methods: Support Vector Learning*, chapter Making large-Scale SVM Learning Practical. MIT Press, 1999.
18. A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/∼mccallum/bow, 1996.

19. H. W. Mewes, K. Albermann, M. Bähr, D. Frishman, A. Gleissner, J. Hani, K. Heumann, K. Kleine, A. Maierl, S. G. Oliver, F. Pfeiffer, and A. Zollner. Overview of the yeast genome. *Nature*, 387:7–65, 1997.
20. H. W. Mewes, K. Albermann, K. Heumann, S. Liebl, and F. Pfeiffer. MIPS: A database for protein sequences, homology data and yeast genome information. *Nucleic Acids Research*, 25:28–30, 1997.
21. H. W. Mewes, J. Hani, F. Pfeiffer, and D. Frishman. MIPS: A database for genomes and protein sequences. *Nucleic Acids Research*, 26:33–37, 1998.
22. H. W. Mewes, K. Heumann, A. Kaps, M. K., F. Pfeiffer, S. Stocker, and D. Frishman. MIPS: A database for protein sequences and complete genomes. *Nucleic Acids Research*, 27:44–48, 1999.
23. D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.
24. T. M. Mitchell. *Machine learning*. McGraw Hill, New York, 1996.
25. J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
26. J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.
27. B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *Proc. ACM Conf. on E-Commerce*, pages 158–167, 2000.
28. B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems–a case study. In *WebKDD 2000 Workshop Notes*, 2000.
29. B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. WWW10*, 2001.
30. M. Schena, D. Shalon, R. W. Davis, and P. O. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270, 1995.
31. V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.
32. V. Vapnik. *Statistical Learning Theory*. John Wiley, New York, 1998.
33. V. E. Velculescu, L. Zhang, B. Vogelstein, and K. W. Kinzler. Serial analysis of gene expression. *Science*, 270, 1995.
34. S. M. Weiss and C. A. Kulikowski. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning and Expert Systems*. Morgan Kaufmann, San Mateo, CA, 1991.
35. D. Wettschereck, D. Aha, and T. Mohri. A review and empirical evaluation of feature-weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11:273–314, 1997.