

Mining Microarray Gene Expression Data

Michinari Momma⁽¹⁾

Minghu Song⁽²⁾

Jinbo Bi⁽³⁾

(1) mommam@rpi.edu, Dept. of Decision Sciences and Engineering Systems

(2) songm@rpi.edu, Dept. of Chemistry

(3) bij2@rpi.edu, Dept. of Mathematical Sciences

Dec. 2nd, 2001

1 Introduction

DNA microarray technology provides biologists with the ability to measure the expression levels of thousands of genes in a single experiment. Many initial experiments suggest that genes of similar function yield similar expression patterns in microarray hybridization experiments. Hence it makes possible to distinguish genes with different functions by analyzing gene data generated from microarray hybridization experiments. As data from such experiments accumulates, it will be essential to have accurate means to extracting biological significance and using the data to assign functions to genes.

Most current approaches in this area attempt to learn functionally significant classifications of genes in an unsupervised fashion because it is inherently a clustering problem that group a bunch of genes with large similarity in gene expression data and hopefully they turn out to have similar function. Unsupervised gene expression analysis methods begin with a definition of similarity between expression patterns, but with no prior knowledge of the true functional classes of the genes. Genes are then grouped by using certain clustering algorithm such as hierarchical clustering [1] or Kohonen maps, i.e., self-organizing maps [2].

Most of the functions of genes are able to be measured based on some biological experiments so that desired function classes can be exactly assigned to genes. Thus the clustering problem is converted to be a classification problem since the difference between them is one is to learn in absence of teacher signals, and the other incorporates prior knowledge to instruct learning. In our project, we apply Support Vector Machines (SVMs) [3], a supervised machine learning technique into gene data. Supervised learning uses a training set with some of training data be members of a functional

class, and others non-members of the class, to specify in advance which data should cluster together.

SVMs offer two primary advantages with respect to unsupervised learning such as hierarchical clustering and self-organizing maps. First, although all three methods employ distance (or similarity) functions to compare gene expression measurements, SVMs are capable of using a larger variety of such functions, so-called kernel functions in high-dimensional feature spaces. This allows the SVMs implicitly to take into account correlations between gene expression measurements. Second, as we talked before, supervised methods like SVMs take advantage of prior knowledge in making distinctions between one type of gene and another.

2 Statement of Problem and Data

We have a synthetic data generator coded in MatLab as included in the tar file, but we prefer to working on a real-life data, which is obtained from a project webpage <http://www.cse.ucsc.edu/research/compbio/genex/expressdata.html> of Munich Information Center for Protein Sequences (MIPS). The definition of gene classes and more data sets can be found in the following MIPS Yeast Genome Database webpage <http://www.mips.biochem.mpg.de/>.

The expression data set consists of 2467 genes from the budding yeast *Saccharomyces cerevisiae* measured in 79 different DNA microarray hybridization experiments. In another words, it has 2467 rows, each of which represents a sample gene by the microarray expression. Each gene or say each sample has 79 attributes, and each of the attributes is generated from one experiment. It means the data has 79 attribute columns. Moreover, from these data, we learn to recognize 6 functional classes, which are TCA, Resp, Ribo, Prot, Hist, HTH. So the data also has 6 class label columns. Recognizing each functional class is a binary classification problem, which means either the gene has the specific function (labelled as +1) or not (labelled as -1).

2.1 Three Characteristics in Data

- As you can see, the data is developed for dealing with 6 functional classes, and each is a binary-labelled classification. It implies being able to build a good classification rule for only one of 6 classification problems may not be enough. However modelling classification rules for 6 functional classes looks pretty much separable if we assume there is no strong interaction among them. We can build one by one 6 classification models using same attribute value matrix and different labels.
- The data set has many missing values. For some of genes, the 79 microarray hybridization experiments may not be totally applicable or some experimental

Table 1: The Summary of Data

Labels	TCA	Resp	Ribo	Prot	Hist	HTH
+1	17	30	121	35	11	16
-1	2450	2437	2346	2432	2456	2451

results are lost, therefore certain genes do not have values for some attributes. Some steps of data preprocessing are indispensable before the normal SVMs experiments start.

- The most difficult problem for SVMs or other techniques is that the data is distributed very unevenly. Table 1 summarizes the numbers of genes in different label columns. This makes the data very tough in learning. We have to say a good accuracy of misclassification rate may not make sense for this data. For example, you could get 90% of data examples are classified correctly, but the model may do nothing but classify all examples as label -1 . Hence some statistics other than misclassification rate should be developed to tackle the skew data.

3 The Mining Methodologies

3.1 Support Vector Machines

Support vector machines have been shown to perform well in multiple areas of biological analysis including evaluating microarray expression data, detecting remote protein homologies, and recognizing translation initiation sites, etc. Let (x_i, y_i) , $i = 1, 2, \dots, N$ be the sample example of genes where x_i is a vector of 79 attribute values and y_i is a scalar binary label ($+1$ or -1). Suppose the classification model is in the hypothesis space $f(x) = \sum_{i=1}^N y_i \alpha_i K(x_i, x) + b$ where the α 's and b are coefficients to be determined. A classic 2-norm support vector machines can be formulated as the following optimization problem in dual form

$$\begin{aligned}
 \min_{\alpha, b} \quad & \frac{1}{2} \sum_{i,j=1}^N y_i y_j \alpha_i \alpha_j K(x_i, x_j) - \sum_{i=1}^N \alpha_i \\
 \text{s.t.} \quad & \sum_{i=1}^N y_i \alpha_i = 0, \\
 & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, N.
 \end{aligned} \tag{1}$$

where the function $K(x_i, x_j)$ is the so-called kernel function which should satisfy the Mercer theorem to be able to replace the inner product of original attribute vectors $\langle x_i, x_j \rangle$. The coefficient b can be determined by using KKT complementarity conditions such as $\alpha_i [y_i (\sum_{j=1}^N \alpha_j K(x_i, x_j) + b) - 1 + \xi_i] = 0$, $i = 1, \dots, N$ where ξ_i is the slackness variables.

We carried out experiments on 3 different versions of SVM classification approaches. We improve the performance of SVMs in terms of speed of convergence. We first tried SVMTorch which is included in our first report, and our new methods Proximal SVM and Reduced SVM have much faster running time although some results may not be better or as good as those by SVMTorch.

3.1.1 SVMTorch

One of the method is well-known SVMTorch which is proposed to efficiently solve large-scale regression problems (more than 20000 examples), but the method can also be applied to classification problems. SVMs are trained by solving a quadratic optimization problem which needs on the order of l^2 memory and time resources to solve. SVMTorch uses a matrix decomposition algorithm and hence yield significant time improvements.

The decomposition algorithm can be divided into 4 steps. First select problem variables as a new working set, and then fix the other variables to their current values and solve the problem with respect to the selected working set. After that, search for variables whose values have been at lower or upper bound for a long time and that will probably not change anymore, then optionally perform shrinking phase to make problem smaller by removing these variables. The last step is to test whether the optimization is finished. If not, return to the first step. The detailed description about SVMTorch can be found in [?].

3.1.2 Proximal SVM (P-SVM)

Instead of a standard support vector machine that classifies points by assigning them to one of two disjoint half-spaces, Proximal SVM classifies points by assigning them to the closest of two parallel planes (in input or feature space) that are pushed apart as far as possible. The formulation in Proximal SVM can also be interpreted as regularized least squares and considered in the much more general context of regularized networks, and hence it leads to an extremely fast and simple algorithm for generating a linear or nonlinear classifier. The classifier merely requires the solution of a single system of linear equations. Computational results indicate [8] that it has comparable test set correctness to that of standard SVM classifiers, but with considerably faster computational time that can be an order of magnitude faster.

The key idea of proximal SVM is to regularize the intercept term in classification or regression models not only the normal or in another words the orientation of the model. As the normal trick in optimization, the constraints can be added directly in the objective function with penalty. Then the problem turns to be an unconstrained optimization problem. It can be solved by finding the solution of linear equations that take the gradient of objective function equal zero. Although the method is proposed for both linear and nonlinear problems (or correspondingly the linear and nonlinear

kernels), it has much more efficient performance for linear ones than nonlinear ones based on our experiments and observations.

3.1.3 Reduced SVM (R-SVM)

The motivation for Reduced SVM comes from the practical objective of generating a non-linear separating surface or hyperplane for a large dataset which requires a small portion of the dataset for its characterization. The difficulty in using nonlinear kernels on large datasets is twofold. First is the computational difficulty in solving the potentially huge unconstrained optimization problem (even though it can be formulated as unconstrained problem as we stated in previous section). It involves the kernel matrix $K(X, X')$ that typically leads to the computer running out of memory even before beginning the solution process. The second difficulty comes from utilizing the resulted formula for separating surface on a new unseen point x .

The justification for the Reduced SVM approach is this. It uses a small random sample \bar{A} of the dataset as a representative sample with respect to the *entire* dataset A both in solving the optimization problem and in evaluating the nonlinear separating surface. This approach can also be interpreted as a possible instance-based learning where the small sample \bar{A} is learning from the much larger training set A by forming the appropriate rectangular kernel matrix $K(A, \bar{A}')$ between the original and reduced sets. This formulation works very well computationally as evidence by the computational results in [9].

3.2 Classification Based on Associations

Besides constructing the SVMs classifiers, CBA (classification rule base on association) approach [6] is also used to generate the classifier for comparison. CBA v2.1 (demo version) was downloaded from professor Bing Liu's web site: <http://www.comp.nus.edu.sg/liub/>. CBA algorithm integrates association mining rules and classification mining rules to build the classifiers. The integration is done by focus on subsets of association rules whose right hand-side is restricted to the class label. These subsets of rules are referred as the class association rules (CARs). The existing association rule mining techniques are adapted to mine all CARs that satisfy the minimum support and minimum confidence constraints.

Let us assume that D is a normal dataset, I is the set of all items in D and Y be the set of class labels. If a data case $d \in D$ contains $X \subseteq I$, a subset of items, then a class association rule (CAR) is an implication of the form $X \rightarrow y$, where $X \subseteq I$, and $y \in Y$. A rule $X \rightarrow y$ holds in D with confidence c if $c\%$ of cases in D that contain X are labelled with class y . The rule $X \rightarrow y$ has support s in D if $s\%$ of the cases in D contain X and are labeled with class y . Ruleitems that satisfy minsup are called frequent ruleitems, while the rest are called infrequent ruleitems.

The CBA algorithm consists of two parts, a rule generator (called CBA-RG), which is based on algorithm Apriori to find association rules, and classifier builder (called CBA-CB). In CBA-RG algorithm, it generates the complete set of CARS that satisfy the user-specified minimum support (minsup) and minimum confidence (minconf) constraints and find all the frequent ruleitems by making multiple passes over the data. Once all the rules are found, then the next objective is to build a classifier from CBAs by selecting the best rules covering the training cases. The details of the CBA algorithm can be found in [6].

4 Implementation

4.1 Data preprocessing

As our analysis of data, we have to preprocess data before really start SVM programs or CBA software because of those missing values. Two ways to handle this have been tried. One way is to merely discard all those rows (corresponding to genes) with missing values, which appears not a good idea because after delete those rows, only 601 genes are left and no any positive labels are retained for some functional classes. So the second way is to replace all missing values for one attribute by the sample mean of that attribute. This way we won't miss any examples. In order to make comparison fair, we use the data with missing value replaced by mean of each column for our three SVMs and CBA.

4.2 Statistics

To be able to weaken the effects of the skewness of data, we'd better change the metric used to evaluate the performance of a classifier. What can be a good statistic to evaluate among different classifiers? The very unbalanced proportion of positive and negative class labels in our skew data has to be taken into account. Bearing this task, Kappa statistic and ROC curve seem promising metrics in model evaluation. We tried both in the learning process of classic SVMs as an evaluation part in model selection. In our three SVMs, we adopt both statistics.

The Kappa statistic measures pairwise agreement of true responses and the prediction, correcting for expected chance agreement. Before give the formula of Kappa, let us describe the definition of TP (true positive), TN (true negative), FP (false positive), and FN (false negative). Assume data have two classes, call them positive class and negative class. If the prediction is positive and the true label is also positive, the point will be counted in TP, so the TP measure how many positive points have been classified correct. The FP measures how many points whose real label are negative but be classified as positive. Similar explanation works for TN and FN respectively. A

general formula can be stated as

$$K_{\text{appa}} = \frac{P(A) - P(E)}{1 - P(E)} \quad (2)$$

where $P(A)$ is the observed proportion of true positive TP and true negative TN and $P(E)$ is the expected proportion of TP and TN, calculated based on the assumption of binomial distribution of TP and TN.

The ROC SVM adopts ROC (Receiver Operating Characteristic curve) as the criteria of model evaluation in model selection. ROC curve is a plot of the true positive rate (sensitivity) against the false positive rate (1-specificity) for the different possible cut points of a diagnostic test. The area under the ROC curve serves the metric, for which a good model has larger area than a bad model. The details for ROC curve can be found in [10].

4.3 Patten search algorithm

To make SVMs work in real-life cases, appropriate values have to be determined for some model parameters also called hyper-parameters of SVMs such as the regularization parameter C , and parameters needed in kernel calculation like σ . This kind of problem can be formulated as model selection. In our model selection, we utilize a technique called pattern search (PS). PS is a direct search method. PS was selected because it is a simple effective optimization technique suitable for optimizing a wide range of objective functions. It does not require derivatives, only direct function evaluation – making the method suitable for problems for which it is impossible or hard to obtain information about the derivatives. The convergence to local minima for constrained problems as well as unconstrained problems has been proven.

4.4 High-level design for SVMs and CBA

Figure 1 shows the high-level design of our entire experiments on SVMs with three different implementation approaches. It can be also regarded as the pseudo-code of our programs. The "Test statistic" and the "Val statistic" in the flowchart imply either Kappa statistic or area of ROC curve.

The most important step in executing CBA program is to prepare the input data. CBA has two types of mining strategies (classification mining and association mining), which uses two types of data format respectively: relational database table and transaction data. In our work, we adopt the relational database table format. It needs two input files: *.data and *.names. These two files must have a common prefix. The data file is the table without the first row i.e., the attribute names with comma as the delimiter. For example,

A1, -1 , good A2, 3.5, bad A1, 800, good A3, 3.5, good

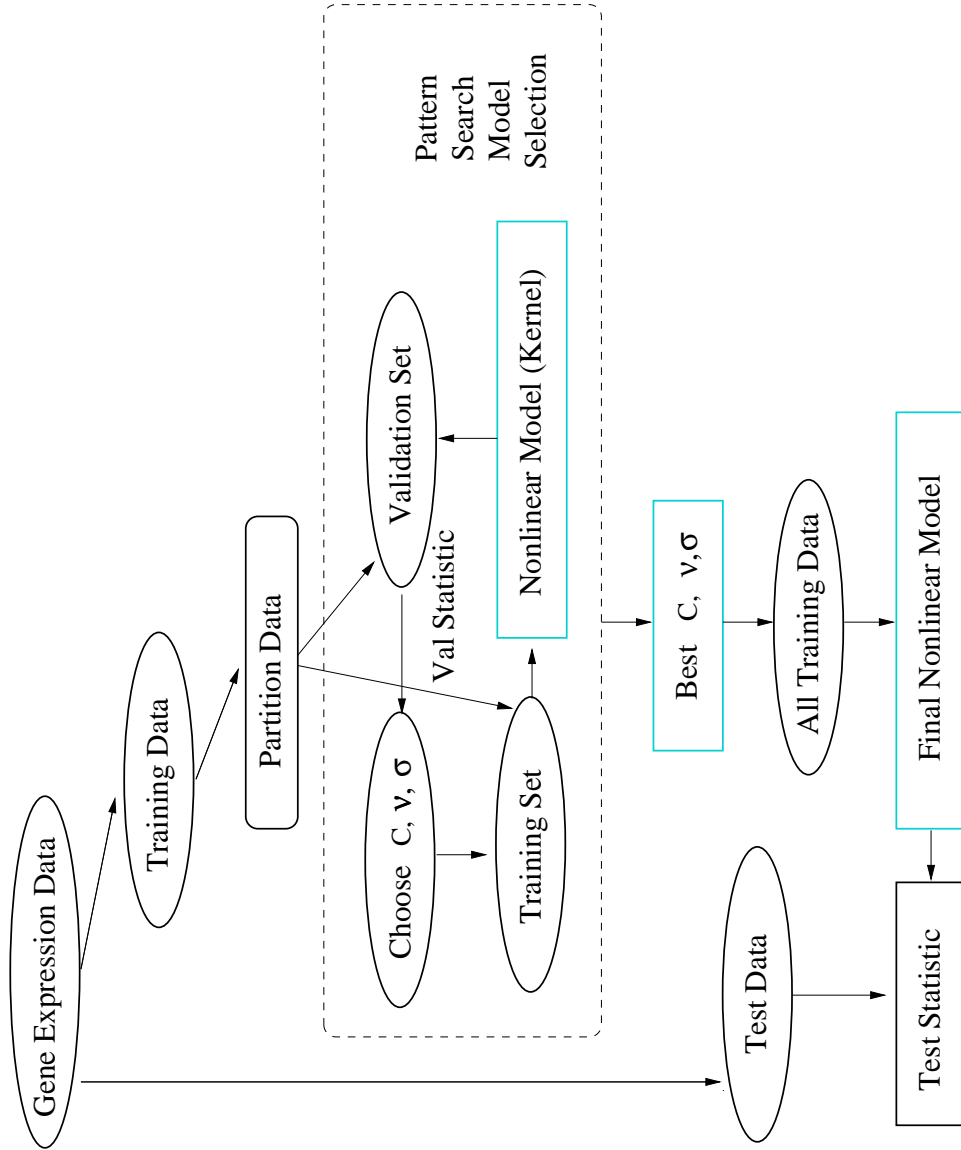


Figure 1: High-level design of the whole process.

Table 2: Summary of error rates for CBA

Class	TP	FP	TN	FN
TCA	0	0	2450	17
Resp	0	0	2437	30
Ribo	92	3	2343	29
Prot	0	0	2432	35
His	0	0	2456	11
HTH	0	0	2451	16

The "names" file format is C4.5 type and its first line has all possible class attribute values. All possible discrete attribute values must be specified in attribute description. The class label is always the last attribute in the data file. If an attribute is a continuous attribute, a keyword "continuous" can be put there. A sample "names" file can be found in the installed directory `\cba\samdata\c45data`.

Since all attributes of microarray gene expression matrix are continuous, so for these continuous attributes, their value range are discretized into intervals and the intervals are mapped to consecutive positive integers. With these mappings, we can treat a data case as a set of (*attribute, integer - value*) pairs, which are called items, and a class label. The auto and manual discretization options have been implemented in the CBA v2.1 program. We could pass our data file with continuous attributes to this system, which will discretize the continuous attributes and output a file that contains the discretized data. Upon return, discretized data can then be used in the next mining step.

4.5 Implementation Details

The executive file "ps_svm" in our zipped file contains three different versions of SVMs and one methodology of model selection. Pattern search is used for parameter tuning of SVMs. For classic SVM, SVMTorch is used. Model selection is based on the cross-validation error estimate. Pattern search and SVMTorch were the core of the previous portion of the project. The main program that combines these methodologies is "ps_svm_1.1.c". For the current project concerned, a function "run_svm_TEST" creates validation sets and pass those validation sets to a function "patternsearch2D". This function implements pattern search for classification.

SVMTorch is one of the fastest SVM implementation approach, but we work on the faster versions of SVM: proximal SVM and reduced SVM in the second portion of the project. These two different versions of SVM are implemented to improve efficiency. The functions are given in "psvr_funcs.c". Depending on which one, "p-svm" or "r-

Table 3: Summary of error rates for two SVMTorch classifiers

Class	Method	TP	FP	TN	FN
TCA	Kappa SVM	2	1	2449	15
	ROC SVM	5	4	2446	12
Resp	Kappa SVM	18	3	2434	12
	ROC SVM	12	5	2432	18
Ribo	Kappa SVM	110	7	2339	11
	ROC SVM	110	8	2338	11
Prot	Kappa SVM	24	5	2427	11
	ROC SVM	11	5	2427	24
Hist	Kappa SVM	9	0	2456	2
	ROC SVM	4	0	2456	7
HTH	Kappa SVM	0	0	2451	16
	ROC SVM	0	0	2451	16

svm” users would like to use, the program calls different functions. See the codes for more detailed information.

5 Experimental Results

We follow the line in [4]. A 3-fold cross validation experiment using our SVM classifiers with either Kappa statistic or ROC curve as measure has been performed. Our experiments show that some functional classes of genes can be recognized by using SVMs trained on DNA microarray expression data. Table 2 presents the numbers of examples as TP, FP, TN or FN in total of 3 folds in the experiments by CBA, which serves our base line. Table 3 depicts the results by SVMTorch with pattern search. The computational results by Proximal SVM and Reduced SVM have been included in Table 4 to show the improvement of the performance especially in terms of the convergence speed. Our results are only slightly worse than those in that paper [4].

CBA can generally be used for multi-classification but the results are much worse compared with SVMs which utilize 6 binary classifiers each for one gene function. So we decide to make a scheme that we can perform 6 binary classifications.

References

- [1] Spellman, P. T., Sherlock, G., Zhang, M. Q., Iyer, V. R., Anders, K., Eisen, M. B., Brown, P. O., Botstein, D. and Futcher, B. (1998) *Mol. Biol. Cell*, 9, pp3273–3297.

Table 4: Summary of performance of P-SVM and R-SVM classifiers

Method/Class	TP	FP	TN	FN	CPU time
SVMTorch Nonlinear					
Ribo	109	10	2336	12	11m38.191s
Prot	24	3	2429	11	
P-SVM					
Ribo	108	6	2340	13	0m51.866s
Prot	0	0	2432	35	
R-SVM					
Ribo	105	5	2341	16	5m42.828s
Prot	4	0	2432	31	

- [2] Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E. and Golub, T. (1999) *Proc. Natl. Acad. Sci. USA*, 96, pp2907–2912.
- [3] Burges, C. J. C., (1998) A Tutorial on Support Vector Machines for Pattern Recognition, *Data Mining and Knowledge Discovery*, Vol. 2, No. 2, pp121–167.
- [4] Brown M., Grundy W., Lin D., Christianini N., Sugnet C., Jr M., and Haussler D. (1999) Support vector machine classification of microarray gene expression data, *UCSC-CRL 99-09*, Department of Computer Science, University California Santa Cruz, Santa Cruz, CA.
- [5] Furey T., Cristianini N., Duffy N., Bednarski N., Schummer M., and Haussler D., (2001) Support vector machine classification and validation of cancer tissue samples using microarray expression data, *Bioinformatics* To appear.
- [6] Liu B., Hsu W., and Ma Y., (1998) Integrating Classification and Association Rule Mining, *Knowledge Discovery and Data Mining*, pp80-86.
- [7] Collobert R. and Bengio S., (2001) SVMTorch: Support Vector Machines for Large-Scale Regression Problems, *Journal of Machine Learning Research*, Vol 1, pp143-160.
- [8] Fung G. and Mangasarian O. L., (2001) Proximal Support Vector Machine Classifiers, *Data Mining Institute Technical Report 01-02, KDD 2001*, San Francisco August 26-29, 2001.
- [9] Lee Y.-J. and Mangasarian O. L., (2000) RSVM: Reduced Support Vector Machines, *Technical report 00-07*, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin
- [10] Kanungo T. and Haralick R., (1995) Receiver operating characteristic curves and optimal bayesian operating points, *Proc. of IEEE Int. Conf. on Image Processing*, Washington, D.C., Vol. 3, pp256–259.