

Online Microarray Analysis Tool using a modified support vector machine (MSVM)

An internship report for CBS MS Degree

Committee:

Dr. Rosemary Renaut¹

Professor, Department of Mathematics and Statistics,
Director, Computational Biosciences PSM
Arizona State University

Dr. Huan Liu²

Professor, Department of Computer Science & Engineering,
Arizona State University

Dr. Hongbin Guo³

Post-doctoral Fellow, Department of Mathematics and Statistics,
Arizona State University

Student:

Wang-Juh Chen (Sting)⁴

Computational Bioscience PSM
Arizona State University

May 9, 2005

Report Number: 05-03

¹email: renaut@asu.edu

²email: Huan.Liu@asu.edu

³email: hb_guo@asu.edu

⁴email: wang-juh.chen@asu.edu

1. Abstract

Microarray is becoming an important tool for monitoring and analyzing gene expression profiles from thousands of genes simultaneously. Due to the characteristic of these datasets, where the dimension of the feature space is far greater than the sample size, we can not use traditional methods for information retrieval. Researchers are looking for some other tools to solve this problem, such as supervised machine learning and data mining. Supervised machine learning and data mining tools are popular for the analysis of gene expression microarray data. The Support Vector Machine (SVM) is one of these. For the non-separable case, SVM introduces slack variables in mapped space as measurements of misclassifications, the source of which may be mislabeling, error of data or/and outliers. Here we investigate a novel way to approach the misclassifications. We account for the misclassifications in feature space by an errors in data approach, based on the observation of the noisy characteristic of microarray data. In this internship, we propose a SVM as a classifier, in which the measurement errors are incorporated in the feature space.

During the training phase, involving huge datasets with large number of features but actual small sample size, it will cost a lot of time and memory, so we will introduce a CLUSTER platform, ROCKS, to distribute computations. Meanwhile, an online submission interface will be included to make this tool easier for access.

The modified SVM algorithm demonstrates the advantages of choosing Support Vectors, doing the regularization, and handling the errors in the dataset. These advantages make the algorithm suitable for different kinds of datasets and tolerant to errors within a dataset, which occurs often during microarray experiments.

Keywords : Microarray, SVM, CLUSTER, LAMP

2. Goals of internship

Develop an online tool to analyze microarray data. Implement a modified support vector machine (MSVM) for classifying microarray data more accurately, and apply CLUSTER system and online submission for the service.

3. Introduction and overview

3.1. Microarray

DNA microarray is a high-throughput tool for rapidly analyzing the expression of all genes within a genome[1,2,3].

With thousands or ten of thousands of genes on the microarray chip, we can determine the expressed genes from a few interesting samples. The most commonly used microarray chip was developed in the laboratory of Patrick Brown at Stanford University, and then sold commercially by the company Affymetrix[1]. The chips are constructed by robotic spotting of DNA fragments, derived from the PCR amplification of entire genes, onto precise points of a glass slide (Figure 1). Consequently, the DNA on the chip is fixed by UV cross-linking and then denatured to make the DNA single stranded. The prepared chips are then used as templates for the binding of labeled cDNA fragments. cDNA

fragments are derived from the RNA or mRNA of interesting target cells. Those samples could be RNA or mRNA which are virtually correlated with all changes in cell state or type. After binding the fluorescents (for example, sample 1 labeled with Cy3, and sample 2 labeled with Cy5) and then the reverse transcription, RNA samples turn into cDNA, which can hybridize with the single strand DNA on the chip (Figure 2). The color appearing on each spot, expresses the ratio of sample 1 and 2 cDNA binding with the single strand DNA of the chip. After the samples are accurately assessed and the fluorescent signals are calculated, the expressed genes under different conditions of a genome-wide scale can be determined.

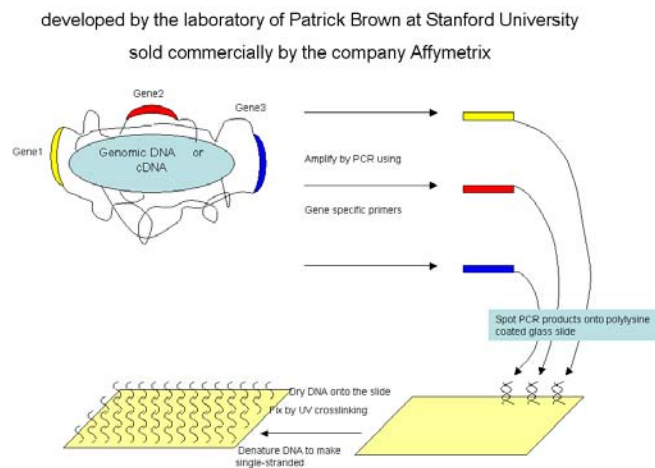


Figure 1 Construction of a DNA microarray

Resource: Richard J Reece. *Analysis of Genes and Genomes*. 2004

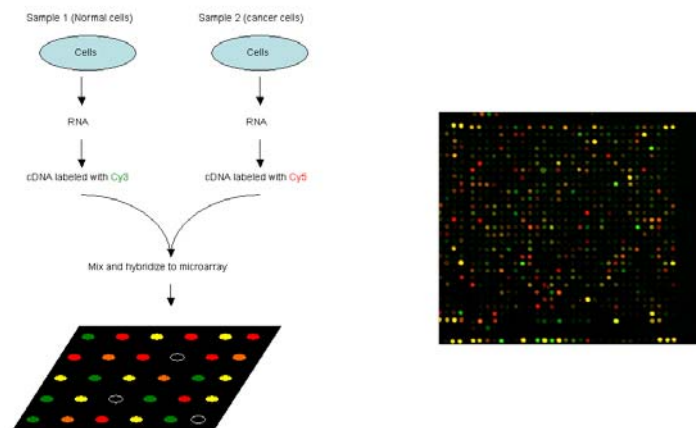


Figure 2 Sample preparation and hybridization with microarray

Resource: Richard J Reece. *Analysis of Genes and Genomes*. 2004

So far, there are lots of biologists using microarrays to do the transcript profiling in many species. For example, DNA microarray has been used to measure the relative expression of all genes in yeast during the diauxic shift. Around 50 percent of the differentially expressed genes identified by microarray analysis had no previously characterized function[1].

Using microarrays is as casting the net in the ocean, you can catch many interesting expressed genes which have not been found or explored in prior research, and eliminate those genes which are not relevant to your topic. Microarray technology is also used in cancer research, stem cell research, and Drug Discovery[1,2,3].

In breast cancer and lung cancer research, microarray analysis has allowed sets of genes to be identified that may be involved in the process of cellular proliferation during cancer growth, through the expression differences between normal and cancerous cells. On the other hand, the use of microarrays in cancer treatment is for the classification of cancerous cells based upon the genes they express. For instance, the assignment of previously unidentified subtypes to malignant melanoma cells can be inferred through their gene expression pattern. These subtypes can be used to predict the clinical outcome of individual melanomas.

There are some stem cell researchers using microarrays to transcript the profiles of different types of stem cells[1]. This has revealed the presence of approximately 200 genes that are expressed within different stem cells that are not expressed within differentiated cells.

Microarray applications include drug Discovery and pharmacogenomics. Identifying drug targets provided the initial market for the microarray[2]. A good drug target has extraordinary value for developing pharmaceuticals. By comparing the differentiations in which genes are expressed in normal and diseased tissue, scientists might be able to identify the genes and hence the associated proteins, or the proteins with which drugs actually interact -- that are part of the disease process. Researchers could then use that information to synthesize drugs that interact with these proteins, thus reducing the disease's effect on the body[2,3]. Microarray analysis can also assist drug companies in choosing the most appropriate candidates for participation in clinical trials for new drugs, through assisting with identification of individuals with similar biological patterns. In the future, this emerging technology has the potential to help medical professionals select the most effective drugs, or those with the fewest side effects, for individual patients.

3.2.Support Vector Machine (SVM)

3.2.1.The introduction of SVM

Neural Networks are a useful and complex branch of mathematics that have been developing for many years. However, their major drawback is the huge quantity of data and the huge numbers of calculations that must be performed on a database of real-world size and significance. These have placed heavy overloads on many computers.

In the late 1970's, Vapnik and others began to develop a specialized branch of Neural Network techniques called Support Vector Machines, or SVM [4]. SVM has been applied to many areas, including pattern recognition of characters, proteomic patterns, etc [4,5,6,7,8]. In the analysis of microarray data, Brown et al., [9], first introduced the SVM for gene expression classification and compared it to four standard machine learning algorithms. Their results show SVM performs better than others. Mukherjee et al, [18], adopted the SVM method for microarray data to generate not only a classifier but also a confidence interval. Furey et al., [19], used SVM in a slightly different way and validated it for three datasets, and presented a careful examination of results for an ovarian dataset in order to show SVM's ability of detecting mislabeling. We also note other methods for classification, for example, SVM recursive feature elimination, [13], partial least squares, [16], singular value decomposition, [11], and the minimum-error distance threshold method, [10].

3.2.2. Standard SVM[5]

To state the problem, we use vector $x^i = (x_1^i, x_2^i, \dots, x_n^i)^T$ to represent the processed n measured gene feature values of sample i , where $(x^i)^T$ is the i^{th} row of X . The whole dataset of m samples is denoted by matrix $X \in \mathcal{X} \subset \mathcal{R}^{m \times n}$. The basic linear SVM approach supposes a given training set $\{x^i, t^i\}_{i=1}^m$ with input data x^i and output data labels $t^i \in \{-1, +1\}$. The purpose is to find a decision function (classifier) which separates the data into two classes. The linear SVM solves the following optimization problem:

$$\begin{aligned} \min_{w, b, \xi} J_p(w, \xi) &= \frac{1}{2} \|w\|_2^2 + \lambda \sum_{i=1}^m \xi^i, \\ \text{subject to } t^i [w^T \phi(x^i) + b] &\geq 1 - \xi^i, \\ \xi^i &\geq 0, i = 1, \dots, m, \end{aligned} \quad (1)$$

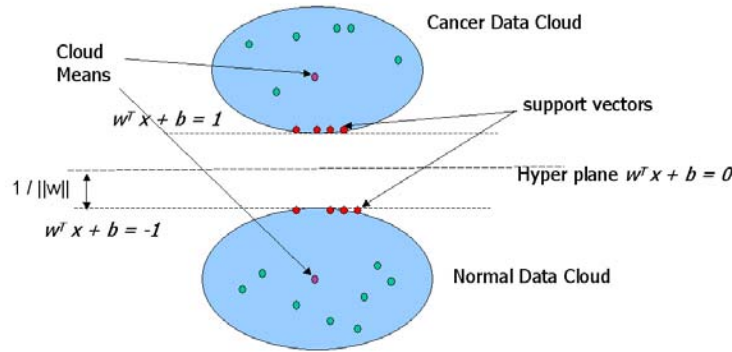


Figure 3 SVM Hyperspace

where ξ^i are *slack variables*, λ is a real positive constant and function ϕ maps data to feature space, $\phi: X \rightarrow H$. In the dual problem a kernel function $K(x^1, x^2) = \phi(x^1) \phi(x^2)$ is introduced, which can be viewed as a measurement of similarity between x^1 and x^2 , yielding the problem

$$\min_{\alpha} \left\{ \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j t^i t^j K(x^i, x^j) - \sum_{i=1}^m \alpha_i \right\},$$

subject to

$$\sum_{i=1}^m \alpha_i t^i = 0, \quad \alpha_i \geq 0,$$

where $\alpha_i, i = 1, \dots, m$, are Lagrange multipliers.

Solving the equivalent quadratic programming problem yields the classifier

$$y(x) = \text{sign} \left[\sum_{i=1}^m \alpha_i t^i K(x, x^i) + b \right]. \quad (2)$$

3.2.3. Linear SVM for noisy data

The objective function of SVM includes *slack variables* ξ^i , which measure the distances of $f(x^i)$ to the margins $f(x)=1$ and $f(x)=-1$ for misclassified samples x^i . SVM minimizes the weighted inverse of the margin and the total squared distances of $f(x^i)$ to the margins, where x^i are misclassified samples.

We are interested in the case when the measured expression data contains large error. Therefore the misclassifications are mainly due to the errors in original data. To model this problem, we propose that the distance be measured in the feature space. In other words, instead of introducing *slack variables* ξ^i , we incorporate the error in data matrix X by an error matrix E . The linear SVM with errors in data is modeled as follows

$$\begin{aligned} \min_{w, b, E} J_p(w, E) &= \frac{\mu}{2} \|w\|_2^2 + \frac{1}{2} \|E\|_F^2, \\ \text{subject to } (A+E)w + bt &\geq e, \end{aligned} \quad (3)$$

where $t = (t^1, t^2, \dots, t^m)^T$, $A = \text{diag}(t)X$, $e = (1, 1, \dots, 1)^T$ and $\|\cdot\|_F$ represents the Frobenius norm. Apparently, the rows of E are zero if and only if the corresponding samples are error free. Positive parameter μ trades off between the margin and estimated errors of the training data. The Lagrangian for (3) is

$$\mathcal{L}(w, \varepsilon, b, \alpha) = \frac{\mu}{2} \|w\|_2^2 + \frac{1}{2} \|E\|_F^2 - \alpha^T [(A+E)w - e + bt]. \quad (4)$$

The dual problem is given by

$$\max_{\alpha \geq 0} \min_{w, E, b} \mathcal{L}(w, \varepsilon, b, \alpha).$$

The minimum with respect to w, ε, b of $\mathcal{L}(w, \varepsilon, b, \alpha)$ is given by

$$\nabla_{\varepsilon} \mathcal{L} = E - \alpha w^T = 0, \quad (5)$$

$$\nabla_w \mathcal{L} = \mu w - (A+E)^T \alpha = 0, \quad (6)$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\alpha^T t = 0. \quad (7)$$

We introduce new weight parameter $\zeta = \frac{1}{\mu - \|\alpha\|_2^2}$ replacing μ , and rewrite the above conditions as follows

$$E = \zeta \alpha \alpha^T A, \quad (8)$$

$$w = \zeta A^T \alpha, \quad (9)$$

$$\alpha^T t = 0. \quad (10)$$

Thus, the dual problem is

$$\min \frac{\zeta}{2} \|A^T \alpha\|_2^2 - \sum \alpha_i, \quad (11)$$

$$\text{subject to } \alpha^T t = 0, \text{ and } \alpha \geq 0. \quad (12)$$

The dual formulation has far fewer variables, especially for the case $m \gg n$, as is the case with microarray data. After solving for α , we can immediately calculate w from (9). To retrieve b , we use the Kuhn-Tucker condition of the primary problem,

$$\alpha_i [(A + E)w + bt - e]_i = 0, \quad i = 1, 2, \dots, n.$$

For each support vector, i.e., $\alpha_i \neq 0$, we have an estimation of b ,

$$b^{(i)} = [\alpha_i - \alpha_i [(A + E)w]_i] / t_i, \quad i = 1, 2, \dots, s,$$

where E and w are calculated by (8) and (9) resp., and s is the number of support vectors.

The estimate of b is then set to the average of $b^{(i)}$, $b = \sum_{i=1}^s b^{(i)} / s$.

On the parameter ζ , we suggest a positive number in $[0, 1]$ based on our tests for microarray data. For the samples satisfying $t^i (Aw + b) > 1$ the Lagrangian parameter α_i is identically zero, $\alpha_i = 0$. The only support vectors are those with $\alpha_i > 0$. For microarray data, α_i is usually very small, less than 0.001.

3.3. Computation platform - CLUSTER

3.3.1. The need for CLUSTER

The most time consuming part of the modified SVM algorithm is doing the validation. In data mining, applying an appropriate N-fold cross validation, e.g. 10-fold cross validation, will not require too much computation since the feature size is far less than the sample size. In our case, with the limitations in obtaining microarray datasets, the sample size is less than 105 while the feature space has size over 7,129[23] (Table 2: the list of dataset used). Therefore, we should choose Leave One Out Cross Validation to increase the number of training sets. In the lymphoma dataset, to execute a full Leave One Out Cross Validation needs 31 sets of training and testing, and the time is 280 seconds¹. When we introduce the CLUSTER distributed computation into our program, the time reduces dramatically to only 26 seconds which is 1/10 of the original. Applying the CLUSTER platform will reduce the time of the whole process and both SVM and the MSVM will be more powerful.

3.3.2. The architecture of CLUSTER platform

The advantage of the CLUSTER platform is to divide a high CPU consuming job into several identical tasks and distribute these tasks among many computers. By doing so, the time for the computation will be reduced. Here is the architecture of CLUSTER platform.

A CLUSTER platform contains four parts (Figure 4)[23]:

Frontend

Nodes of this type are exposed to the outside world. Many services (NFS, DHCP, NTP, MySQL, HTTP, ...) run on these nodes. In general, this requires a competent systems administrator. Frontend nodes are used for users login, submission of jobs, code compilation etc.. They can also act as a router for other CLUSTER nodes by using network address translation (NAT).

Frontend nodes generally have the following characteristics:

1. Two Ethernet interfaces - one public, one private.
2. Lots of disk space to store files.

Compute nodes

These are the workhorse nodes. These nodes are not seen on the public Internet. Compute nodes have the following characteristics:

1. Power Cable
2. Ethernet Connection for administration
3. Disk drive for caching the base operating environment (OS and libraries)
4. Optional high-performance network (e.g., Myrinet)

Ethernet Network

All compute nodes are connected with Ethernet on the private network. This network is used for administration, monitoring, and basic file sharing.

Application Message Passing Network

All nodes can be connected with Gigabit-class networks and required switches. These are low-latency, high-bandwidth networks that enable high-performance message passing for parallel programs.

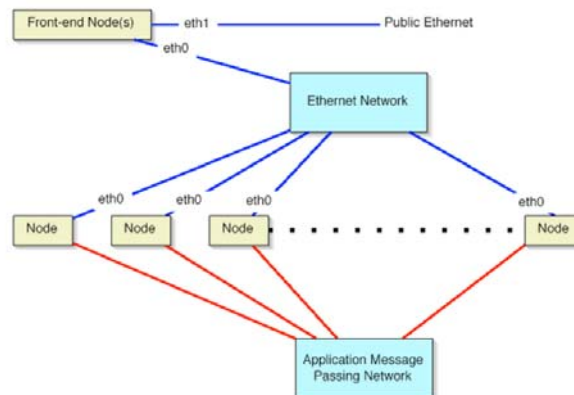


Figure 4 The architecture of CLUSTER platform(ROCKS)

Resource: <http://www.rocksCLUSTERS.org/Rocks/>

3.3.3.MATLAB Distributed Computing Toolbox and Engine

In addition to the hardware, we need the application to implement our algorithm. Message Passing Interface (MPI) is the standard for implementing programs on multiple processors. MPI defines C and Fortran language functions for doing point-to-point communication in a parallel program. MPI has proven to be an effective model for implementing parallel programs and is used by many of the world's most demanding applications (weather modeling, weapons simulation, aircraft design, etc.).

MatlabMPI is set of MATLAB scripts that implement a subset of MPI and allow any MATLAB program to be run on a parallel computer [22].

Similar to MatlabMPI, MATLAB Distributed Computing Toolbox and the Distributed Computing Engine allow users to coordinate and execute independent MATLAB operations simultaneously on a CLUSTER of computers, speeding up execution of large MATLAB jobs.

Figure 5 shows a basic distributed computing configuration. Table 1 is the description of MATLAB distributed computing terms. The procedure for using the distributed computing toolbox is as follows:

1. Find a Job Manager : Your network may have one or more job managers available. The function you use to find a job manager creates an object in your current MATLAB session to represent the job manager that will run your job.
2. Create a Job : You create a job to hold a collection of tasks. The job exists on the job manager, but a job object in the local MATLAB session represents that job.
3. Create Tasks : While your job is in the pending state, you can create tasks to add to the job. Each task of a job can be represented by a task object in your local MATLAB session.
4. Submit a Job to the Job Queue for Execution : When your job is completely defined with all its tasks, you submit it to the queue in the job manager. The job manager distributes your job's tasks to its workers for evaluation. When all of the workers are completed with the job's tasks, the job manager moves the job to the finished state.
5. Retrieve the Job's Results : The resulting data from the evaluation of the job is available as a property value of each task object.

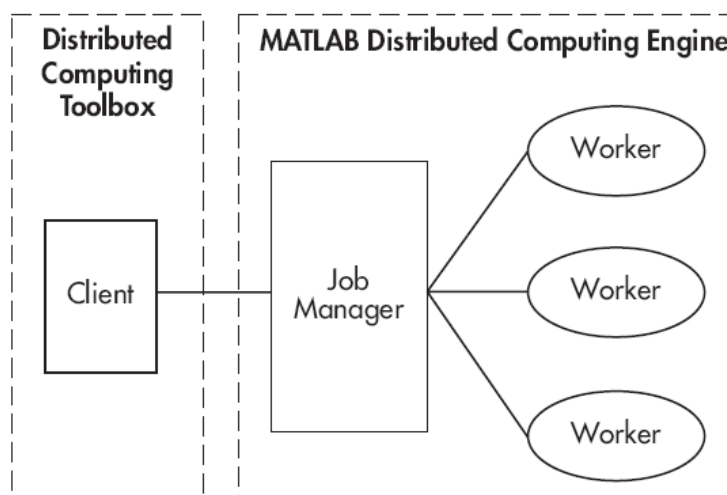


Figure 5 Basic Distributed Computing Configuration

Resource: © 1994-2005 The MathWorks, Inc

Name	Description
Job	The complete large-scale operation to perform in MATLAB, composed of a set of tasks.
Task	One segment of a job to be evaluated by a worker.
Client	The MATLAB session that defines a job using the Distributed Computing Toolbox.
Job manager	The part of the MATLAB Distributed Computing Engine that coordinates job execution, distributing tasks to individual workers for evaluation. This is represented in the client session by a job manager object.
Worker	The session of MATLAB in the MATLAB Distributed Computing Engine that evaluates tasks by executing the tasks' functions. This is represented in the client session by a worker object.

Table 1 MATLAB Distributed Computing Terms

Resource: © 1994-2005 The MathWorks, Inc.

In this internship, we used a CLUSTER platform with one frontend and 18 compute nodes¹. Within the CLUSTER, we installed both MATLAB Distributed Computing Engine and Toolbox on the CLUSTER frontend and MATLAB Distributed Computing Engine on 15 compute nodes. For running the codes in distributed way, we created a job manager on the frontend and 30 workers (2 workers for each node) on the compute nodes.

3.4.LAMP - *Linux-Apache-MySQL-Php*

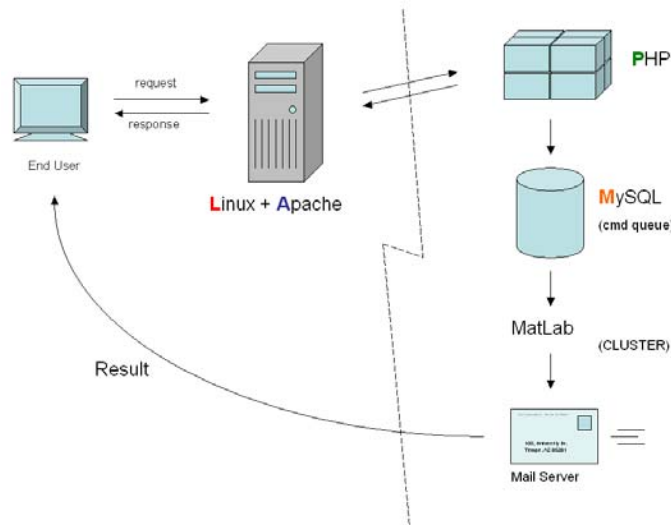


Figure 6 The Structure of The Application

On line applications are the trend in program development, and make it easier for the programmer to publish the software throughout the world. NEOS, for example, developed by Hans Mittelmann et al.[27], is one of the best on line services providing users the ability to solve all kinds of optimization problems remotely. GIST[28], a web base SVM software, was built by William Stafford Noble and Paul Pavlidis and it offers several options for users to choose when they analyze their dataset.

With the goal of providing a service like NEOS and GIST, we have to make an environment to deploy our program. LAMP, which is short for Linux-Apache-MySQL-PHP, is a free package and allows programmers to develop server-side programs. In this internship, we propose an application

which furnishes a mechanism for users to submit their dataset on line and then execute the program through the website. Furthermore, users can choose different SVM packages (e.g. MSVM or SVM) and apply different cross validation (Leave One Out or N-Folds) by clicking the selections provided by our web page.

The advantages of this application are as follows:

1. Easy to use : users don't have to learn how to use MATLAB. For them, running this application is as easy as writing an email on the web base mail.
2. Easy to expand : in this application, we can analyze the same dataset with different SVM algorithms and provide the results for users to compare. These algorithms (packages) can be implemented and added to this application.
3. Cost effective : for the end users, they don't have to buy MATLAB license to run this application.
4. No need for updating : the web service mechanism guarantees to provide the latest version of the application.
5. Cross platform : the interface for this application is the browser, so it allows users to access it from various platforms.

4. Significance

In this internship, we created a new SVM algorithm to analyze microarray datasets and implemented it into a web base application along with the CLUSTER system. The significance of each part is as follows:

1. MSVM : Assumes noise in data measurement (feature space), and provides another way to extract significant genes (feature selection).
2. CLUSTER : Reduces time during the validation part. Easy to implement.
3. LAMP : Cross-platform and User-Friendly.

5. Datasets

	Gene Number	Cancer	Normal	Sample Size
lymphoma Training	7129	11	27	38
lymphoma Testing	7129	14	20	34
ovarian	87558	14	17	31
myeloma	7129	74	31	105

Table 2 The datasets used in this internship

We collected 3 datasets from different public web sites. The gene number, the number of class level, and the sample size are shown in Table 2. For lymphoma dataset, it had already been divided into the training set and testing set.

lymphoma : These datasets contain measurements corresponding to ALL and AML samples from Bone Marrow and Peripheral Blood[12]. There are two datasets containing the initial (training, 38 samples) and independent (test, 34 samples) datasets.

ovarian : The experiments are performed using 97,802 DNA clones, each of which may or may not correspond to hundred genes, for 31 tissue samples[9]. These samples are either cancerous

ovarian tissue, normal ovarian tissue, or normal non-ovarian tissue. For the purpose of these experiments, the two types of normal tissue are considered together as a single class.

myeloma : The myeloma dataset is a new, publically-available cancer data set consisting of 105 highly purified plasma cell samples, from 74 newly diagnosed cancer samples and 31 normal healthy donors[23]. The cancer being studied is multiple myeloma, an incurable malignancy of immunoglobulin secreting plasma cells that grow and expand in the bone marrow.

6. Methods and Experiments

In order to show the advantage of our proposed algorithm, MSVM, we compare with the SVM written by Steve Gunn[29]. Thus in all cases when referring to results using SVM, this implies his algorithm. All the datasets in this internship will be trained and tested by these two methods.

6.1. Data preprocessing

Before we apply any analysis, we have to do data preprocessing. The reasons to do data preprocessing in microarray analysis could be following:

1. For some research, you may know that some genes have no relationship with this disease. So these genes could be removed before the analysis.
2. There maybe exist some outliers within the datasets and these will effect the result. Sometimes, removing these outliers will increase the accuracy.
3. The range of the gene expression could be different from slide to slide. Choosing the right normalization could reduce the differences across datasets and eliminate artifacts.

Here, we applied normalization on our dataset by dividing all the values by the array mean.

$$X_j = \frac{X_j}{\bar{X}}$$

6.2. Error in the dataset

We incorporate the error in data matrix X by a error matrix E . In order to assess the impact of error, we perturbed them by different ϵ , to give datasets

$$X = (1 + \epsilon * randn) X.$$

If we used this formula to generate the perturbed datasets for each analysis, it would lead to evaluation on nonequal datasets. Thus, we did each perturbation only once and saved these datasets into different files, e.g. ovarian_0.5.txt means the original dataset perturbed with $\epsilon = 50\%$.

6.3. Gene selection

In the MSVM, we calculated the weights for each set of genes (features) by (9). The bigger the weight, the more important its contribution to the target. We called the method in which genes are selected based on their weights Wsel.

At the same time, we also introduced another method called F-scores[19]. For each gene x_j , we calculated the mean μ_j^+ (resp. μ_j^-) and standard deviation σ_j^+ (resp. σ_j^-) using the sample labeled +1 (resp. -1). Then we can get the F-score for each gene as

$$F(x_j) = \left| \frac{\mu_j^+ - \mu_j^-}{\sigma_j^- + \sigma_j^+} \right| \quad (13)$$

The F-score with highest number indicates that the expression levels differ most on average. Genes selected based on the largest F-score values give the method Fsel.

For the number of significant genes, we set up different numbers, e.g. 50, 100, 1000, and use these numbers as our significant gene numbers. Thus, for Wsel and Fsel, we sort the weights and F-scores respectively and choose those with highest values, e.g. 100 of these rank ordered, as the significant genes. These genes will be the only features in the datasets during the analysis.

6.4. All, Wsel, and Fsel

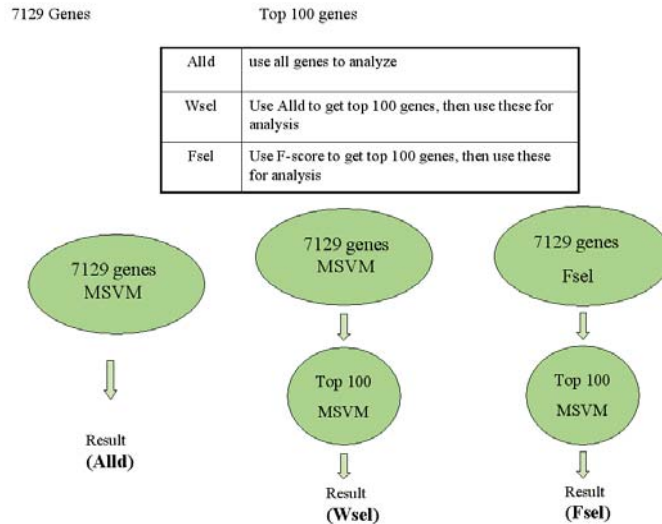


Figure 7 All, Wsel, and Fsel

In addition to Wsel and Fsel, we also used all genes during the analysis and we called this method All. The process of applying these methods is shown in Figure 7. For All, we used all genes for analysis and then calculated the error rate. For Wsel, we used All method to generate a list of top genes (with higher weights) and then applied these top genes to do the analysis. For Fsel, we used the genes with higher F-scores and then only applied this subset of genes for the analysis.

6.5. Regularization

Both of the MSVM and SVM require a term to do the regularization during calculation, but there is a major difference. In order to prevent problems when the Hessian occurring in the SQP is badly conditioned, we have to add a small amount of zero order regularization. In the SVM package, a constant term (10^{-10}) is added to the Hessian matrix and it works well in all but one case in our

experiments. On the contrary, the value we add into the Hessian matrix depends on $\|H\|_2$, so it works in all conditions. The Hessian matrix after regularization is calculated by

$$\tau = 10^{-5} \times \|H\|_2, \tag{13}$$

$$H = H + \tau \times I. \tag{14}$$

6.6. Choosing Support Vectors

One of the advantages of SVM is that it can choose a proper set of Support Vectors to represent the whole dataset for the future analysis. How to set a precise threshold to gain the Support Vectors will influence the accuracy during the testing phase. In our algorithm, we calculated the threshold based on the distribution of all the candidate Support Vectors, while SVM applies a constant value. Our threshold can be calculated by

$$\begin{aligned} \text{Support Vectors} &\in \{\alpha, |\alpha| \leq \delta\}, \\ \text{where } \delta &= 0.01 \times \max(|\alpha|). \end{aligned} \tag{15}$$

6.7. Outlier and mislabeled data

For the *slack variables* ξ which represent distances of $f(x)$ to the margins $f(x)=1$ and $f(x)=-1$ for misclassified samples, we believe that our algorithm could be more reliable when error exists. One of our original datasets includes one mislabeled and one outlier point. This was used to generate three datasets: 1. the original dataset, 2. the same as the original one but change the mislabeled into correct one, and 3. the dataset with the correct label and also remove the outlier. With these datasets, we can apply our algorithm to see how well it classifies when not contaminated by outlier and mislabeled samples.

6.8. The application

The flow chart of the whole process is shown in Figure 8. Initially, the microarray samples were stored separately. One file represents one sample and stored in specific format (e.g. Affymatrix format). We use Perl to combine them into one single file with class labels in the first column. Each

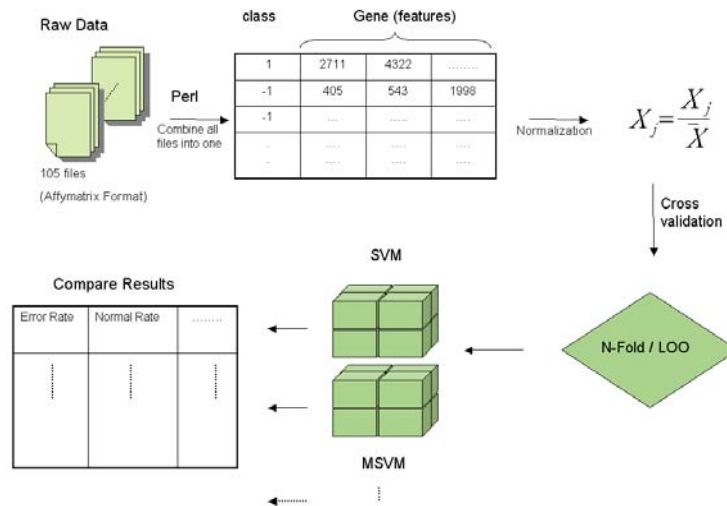


Figure 8 Flow Chart of The whole Process

row stands for one dataset, so the row number is the same as the sample size and the column number is the same as the gene number.

Before the analysis, we applied the normalization on the whole dataset such that the mean of each sample (slide) is 1. When we do the analysis, we have to decide what kind of cross validation we want to use. For our case, we chose Leave One Out Cross Validation since our sample size is too small and we want to get as large a training dataset as possible.

After we generated our training and testing datasets, we used both SVM and MSVM. The error rate for both algorithms was calculated with the same training and testing dataset. In the end, we output a table to compare the algorithms with different conditions.

The flow chart of the application is shown in Figure 9. This web base application allows users to upload their own datasets and setup all the parameters they wish for the analysis and then submit the job. Before they submit, they can choose to perform this job in foreground or background. If their dataset is very small, the foreground option provides an interactive mode for users wishing to submit the job and receive the result through browser. For the larger datasets, we suggest users to choose the background option which allows them to execute their job without the internet connection and receive the results through email. During the computation, we introduced the distributed MATLAB engine so it will save lots of time.

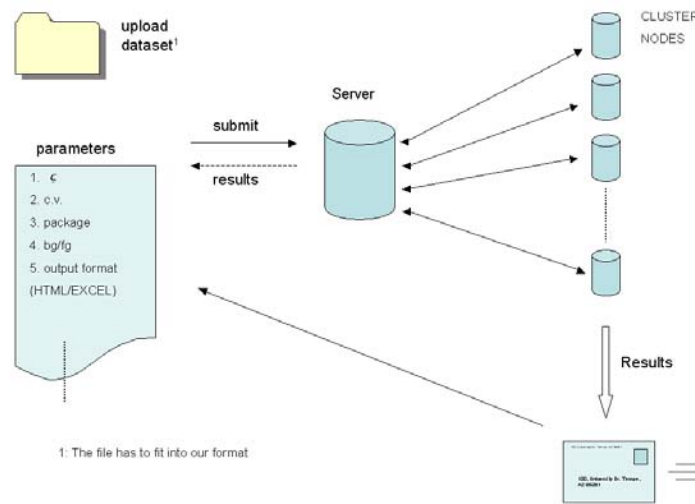


Figure 9 Flow Chart of The Application

7. Results and Discussion

7.1. Choosing the Support Vectors

Method	GenNo	ERate	SGERate	normW	normE
AllId	87558	29.00%	64.50%	3.13E-002	6.28E-005
Fsel	1000	16.10%	16.10%	1.62E-001	1.43E-003
Wsel	1000	32.30%	32.30%	7.68E-002	1.22E-004

Table 3 The error rate of ovarian dataset with $\zeta = 1$ and LOO C.V.

In Table 3, we analyze the ovarian dataset with different gene selection methods and both SVM packages. Here GenNo is the numbers of genes used, ERate is the error rate using MSVM, SGERate is the error rate using Steve Gunn's SVM and normW and normE are norms of w and E , respectively for MSVM. Both packages get the same error rates except in All. If we look at the proportion of the dataset, where 54.84% belongs to normal, we can find that it is very easy to get an error rate less than 54.84% when all the predictions belong to one class. Due to the nature of this dataset, for which the samples can not be linear-separable, the best error rate we can get by using linear SVM is 26% [19]. This result suggests that both methods MSVM and linear SVM are comparable since there is only one sample difference between 26% and 29%. But how could SVM yield an error rate larger than 54.84%? The reason is in the process of choosing the Support Vectors. The candidate Support Vectors for both SVM and MSVM are similar but the selected Support Vectors in each case are totally different. The problem is caused by the threshold for choosing Support Vectors.

The range of candidate Support Vectors is from 0.641×10^{-4} to 0.25×10^{-5} and four of them are less than 10^{-5} . In the MSVM algorithm, the threshold depends on 1% of the max candidate Support Vector, in this case, all of them will satisfy the criterion since the threshold is 0.641×10^{-6} . The difference between these two sets of Support Vectors impacts their predictive ability in the testing phase.

7.2. Regularization

In addition to microarray datasets, we also apply both SVMs on some other datasets in which the number of features is larger than the sample size. Ionosphere dataset [25] is one of them. With 35 features, the dataset includes 225 instances belonging to class I and 126 instances belonging to class II. The error rates of SVM and MSVM are 33.0% and 31.9% respectively, and the number of instances for 1.1% difference is 4. To determine the reason, we check the difference when both methods apply the regularization. The diagonal values of the Hessian matrix for this dataset are between 2.99×10^6 and 0 while $\|H\|_2 = 5.70 \times 10^6$. In order to do the regularization, we have to add a small amount. But in this case, the Hessian matrix is too big with respect to 10^{-10} and it will not become regularized with this value. So, we add a proportion of $\|H\|_2$, which is $10^{-5} \times \|H\|_2 = 57.04$ in this case, and it works all the time no matter how big the Hessian matrix.

7.3. Error in datasets - ϵ

Epsilon	GenNo	ERate	SGERate	normW	normE
0.00%	7129	2.80%	9.70%	3.93E-002	9.83E-006
5.00%	7129	2.80%	11.10%	3.93E-002	9.79E-006
50.00%	7129	8.30%	22.20%	3.58E-002	7.14E-006

Table 4 The error rate of lymphoma dataset with All, $\zeta=1$ and LOO C.V.

Our method yields an acceptable result when error ϵ is introduced in the data, see Table 4, which shows error rates for the lymphoma dataset with All and $\zeta=1$. Initially, we apply the original dataset, where $\epsilon=0$, the error rates of MSVM and SVM are 2.80% and 9.70% resp.. But when we increase ϵ to 5%, which could happen during experiments, MSVM keeps the same error rate while for SVM it is raised to 11.10%, which is 1 more misclassification. Furthermore, raising ϵ to 50%, which is an abnormal situation, our error rate becomes 8.30%, which is 4 more

misclassifications, while SVM produces 8 more misclassifications and yields the error rate of 22.20%. This suggests that our algorithm could deal with datasets that are more contaminated by error.

7.4.Outlier and mislabeled data

	Sample Size	Cancer	Normal	Alld	Wsel (100 genes)
Correct Mislabeled + Remove outlier	30	15	15	9/9	2/9
Correct Mislabeled (with outlier)	31	15	16	9/9	0/9
Original (mislabeled + outlier)	31	14	17	9/9	1/9

Table 5 The class proportion of ovarian dataset in different cases and the number of times in which MSVM outperforms SVM

The MSVM outperforms SVM not only when there exists some errors within the datasets but also when the dataset contains outliers or/and mislabeled data. The ξ variables account for the distances from the margin for the samples which are mislabeled and reduces the effect during the training phase. Table 5 shows the results for the dataset with all the correct labels and the outlier removed. Half of the class belongs to cancer. MSVM gets better results in all cases in Alld and 2 out of 9 in Wsel with 100 genes. Then we add the sample with the outlier into the dataset and apply both algorithms again. MSVM still outperforms SVM in all cases in Alld but yields the same error rate in Wsel. The last case is the original dataset, which includes one mislabeled and one outlier. MSVM is again better in all cases in Alld, and 1 out of 9 cases in Wsel with 100 genes. This suggests that MSVM could perform better even if there exists outliers when we apply all genes for analysis. Again this is because the SVM uses a bad threshold to eliminate the candidate Support Vectors causing a poor prediction during the testing phase.

7.5.Overall performance

Total cases	MSVM outperformance	SVM outperformance	Equality
288	103	10	175

Table 6 The overall performance

During the experiments, we design 288 different training and testing cases and apply leave one out cross validation to get the results. Overall MSVM outperforms SVM. In 103 cases where MSVM performs better, the majority are from the ovarian dataset, and the better performance is due to the Support Vectors. In 10 cases where SVM performs better, most of the cases are from the Ionosphere dataset, which can not be linear-separable but for which we still use the linear kernel for both SVM. In over half of the cases, both methods yield the same results.

7.6.Gene selection

Although we propose a way to do the gene selection by adopting the higher weights, it doesn't work well enough during the experiments. The main reason is not the way we define the significant genes but how to determine the number. The number of significant genes used for these experiments is based on the experience rather than theory. By doing so, we may lose some significant genes which

should be included to provide enough information for the analysis. For example, it may be that we need to closely consider whether the weights decrease gradually, or are there clear points that distinguish a set of large weights from a set of small weights?

7.7. CLUSTER

Dataset	Sample Size	File Size (MB)	Time without CLUSTER (sec)	Time with CLUSTER (sec)	Speedup
Lymphoma	72	9.9	279.190	25.950	10.76
Ovarian	31	45.1	173.880	84.850	2.05
Myeloma	105	11.7	274.420	43.750	6.27

Table 7 Performance with and without CLUSTER

The final step was to transfer the program into a CLUSTER platform with the MatLab Distributed Computing Engine and compare the time cost². Table 7 shows the performance with and without CLUSTER when we analyze three datasets. In some cases, e.g. lymphoma dataset, the time difference could be over 10 fold where we don't even try to tune the performance within the codes for the CLUSTER. The CLUSTER version could reduce the time during the cross validation and make our application more competitive.

8. Conclusions

The application of microarray technology could be very useful for a variety of medical research areas. Combining data mining, such as SVM, will improve the accuracy of the analysis and speed up the whole process. In this internship, our MSVM algorithm demonstrates the advantages of choosing Support Vectors, doing the regularization, and handling the errors in dataset. These advantages will be suitable for different kinds of datasets, including those with higher error as occurs often during microarray experiments.

However, how to determine the significant gene number is very a crucial part during the analysis and this is also the area in which we should focus more. If the purpose of the analysis is not prediction of the class but finding the genes, our method may not be a good solution. To solve this problem, we can combine SVM with other feature selection methods such as FCBF, Fast Correlation-Based Filter solution, from Huan Liu and Lei Yu[26]. Here, within the sorted weights, a method is used to filter out the most significant genes by searching the critical weight which separate the weights into significance and non-significance.

9. Future work

Focus on how to determine the significant gene number and publish the application to make this service available on the website so that we can get more comments from others and improve our algorithm.

10.Acknowledgement

I would like to thank Dr. Renaut who helps me not only in this internship but also my academic life. Dr. Guo, who developed the main program, helped me to enhance my mathematic ability. Dr. Liu leads me into the data mining area. Dr. Mittelmann gave me the idea of creating this application. And Renate Mittelmann teaches me how to install CLUSTER platform and MatLab Distributed Computing Engine. Of course, the most important person is my wife. Without her, I don't know how could I accomplish this internship.

11.Note:

11.1.The specification of the CLUSTER : The machine is managed by the computing support group from the Department of Mathematics and Statistics, Arizona State University.

System Server

Server machine is the gateway between the external network and the internal compute nodes. Server node has:

- ◆ AMD Opteron(TM) processor model 250 dual CPUs
- ◆ 2.40 Ghz Processor internal clock speed
- ◆ 1MB Internal L2 Cache Size
- ◆ 8 x 1GB DDR-333 PC2700 RT-ECC Memory
- ◆ Tyan Thunder Server MotherBoard
- ◆ LSI Logic / Symbios Logic 53c1010 Ultra3 SCSI Adapter
- ◆ 3ware Inc 3ware 7000-series ATA-RAID
- ◆ 2 x 250GB 8MB 7200 RPM SATA-150 Hard Disk

Compute Nodes

Each node has:

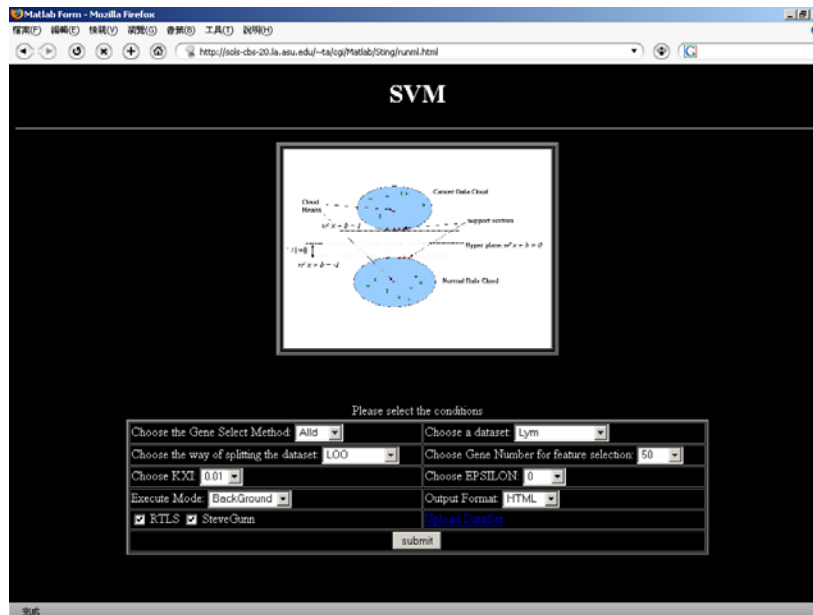
- ◆ AMD Opteron(TM) processor model 250 dual CPUs
- ◆ 2.40 Ghz Processor internal clock speed
- ◆ 1MB Internal L2 Cache Size
- ◆ 8 x 1GB DDR-333 PC2700 RT-ECC Memory
- ◆ AMD 8113 Server MotherBoard
- ◆ 36.4GB 8MB 10000RPM U320 SCSI Hard Disk

Raid Disk

A raid system is used to store user files. It has five 250GB SATA-150 hard disks in an Aries 12-bay U320 SATA system. This is connected to the System Server using a LSI Logic / Symbios Logic 53c1010 Ultra3 SCSI Adapter.

11.2.For the CLUSTER, we used 7 codes to do the computation.

11.3. Snapshot of the application



12. References

- [1]. Richard J Reece. Analysis of Genes and Genomes. 2004.
- [2]. <http://www.geocities.com/bioinformaticsweb/microarrays.html>
- [3]. DNA Microarrays (Genome Chips) (by Leming Shi, PhD). <http://www.gene-chips.com/>
- [4]. Christopher J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery 2, 121-167, 1998. <http://aya.technion.ac.il/karniel/CMCC/SVM-tutorial.pdf>
- [5]. Steve R. Gunn, Support Vector Machines For Classification and Regression, Technique report, 10 May 1998. <http://www.ecs.soton.ac.uk/~srg/publications/pdf/SVM.pdf>
- [6]. J.A.K. Suykens and J. Vandewalle, Least squares support vector machine classifiers, Neural Processing Letters, 9(3), 293-300,1999.
- [7]. Emanuel F Petricoin III, Ali M Ardekani, Ben A Hitt, Peter J Levine, Vincent A Fusaro, Seth M Steinberg, Gordon B Mills, Charles Simone, David A Fishman, Elise C Kohn, Lance A Liotta, Use of proteomic patterns in serum to identify ovarian cancer, Lancet 2002; 359: 572-77
- [8]. Asa Ben-Hur and Douglas Brutlag, Remote Homology Detection: a motif based approach. Bioinformatics, Vol. 19 Suppl. 1 2003, pages i26-i33 http://0-bioinformatics.oupjournals.org.library.lib.asu.edu/cgi/reprint/19/suppl_1/i26.pdf
- [9]. M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. Sugnet, T. S. Furey, Jr. M.Ares, and D. Haussler, Knowledge-based analysis of microarray gene expression data using support vector machines, PNAS 97 (2000), no. 1, 262-267.
- [10]. Li CS and Cheng C, Stable classification with applications to microarray data, Computational Statistics and Data Analysis (2004).
- [11]. Ghosh D., Singular value decomposition regression models for classification of tumors from microarray experiments, Pac. Symp. Biocomput. (2002).
- [12]. Tamayo P Huard C Gaasenbeek M Mesirov JP Collier H Loh ML Downing JR Caligiuri MA Bloomfield CD Lander ES Golub TR, Slonim DK, Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, Science (1999).
- [13]. S. Barnhill I. Guyon, J. Weston and V. Vapnik, Gene selection for cancer classification using support vector machines, Machine Learning (2002).

- [14]. Byrne MC Follettie MT Gallo MV Chee MS Mittmann M Wang C Kobayashi M Horton H Brown EL. Lockhart DJ, Dong H, Expression monitoring by hybridization to high-density oligonucleotide arrays, Nat. Biotechnol. (1996).
- [15]. D. Botstein M. Eisen, P. Spellman and P. Brown., CLUSTER analysis and display of genomewide expression analysis, Proceedings of the National Academy of Sciences.
- [16]. D.V. Nguyen and D.M. Rocke, Tumor classification by partial least squares using microarray gene expression data, Bioinformatics (2002).
- [17]. JP Mesirov D Slonim A Verri T Poggio S Mukherjee, P Tamayo, Support vector machine classification of microarray data, Tech. report, MIT, 1999.
- [18]. Davis RW Brown PO. Schena M, Shalon D, Quantitative monitoring of gene expression patterns with a complementary dna microarray, Science (1995).
- [19]. Nello Cristianini David Bednarski Michel Schummer Terrence S. Furey, Nigel Duffy and David Haussler, Support vector machine classification and validation of cancer tissue samples using microarray expression data, Bioinformatics (2000).
- [20]. Stanford: functional genomics facility. <http://www.microarray.org/sfgf/jsp/home.jsp>
- [21]. The MathWorks - MATLAB Distributed Computing Engine - Execute independent MATLAB algorithms and Simulink models in a computer cluster.
<http://www.mathworks.com/products/distriben/index.html>
- [22]. MIT Lincoln Laboratory - Parallel Programming with MatlabMPI.
<http://www.ll.mit.edu/MatlabMPI/>
- [23]. Rocks Cluster Distribution. <http://www.rocksCLUSTERS.org/Rocks/>
- [24]. D. Page, F. Zhan, J. Cussens, M. Waddell, J. Hardin, B. Barlogie, and J. Shaughnessy, Jr., Comparative Data Mining for Microarrays: A Case Study Based on Multiple myeloma. Technical Report 1453, Computer Sciences Department, University of Wisconsin, Nov 2002.
www.cs.wisc.edu/~dyer/cs540/hw/hw1/ismb02-myeloma.pdf
- [25]. UC Irvine Machine Learning Repository. <http://www.ics.uci.edu/~mlearn/MLRepository.html>
- [26]. Lei Yu, Huan Liu. Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution. Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003. <http://www.public.asu.edu/~7Ehuanliu/papers/icml03.pdf>
- [27]. NEOS Server for Optimization. <http://www-neos.mcs.anl.gov/neos/>
- [28]. GIST SVM Server, <http://microarray.cpmc.columbia.edu/gist/>
- [29]. MATLAB Support Vector Machine Toolbox Version 2.0 Aug. 1998 by Steve Gunn.
<http://www.isis.ecs.soton.ac.uk/resources/svminfo/>