Mining Microarray Expression Data for Classifier Gene-Cores

Goutham Kurra
Dept of ECECS
University of Cincinnati
gkurra@ececs.uc.edu

Wen Niu
Dept of ECECS
University of Cincinnati
wniu@ececs.uc.edu

Raj Bhatnagar
Dept of ECECS
University of Cincinnati
Raj.Bhatnagar@uc.edu

ABSTRACT

We present algorithms and methods for mining DNA microarray expression data from a classification perspective. We demonstrate these algorithms on a gene expression dataset containing two acute leukemia classes. A prioritized feature-selection approach is followed to account for incomplete knowledge of gene function and complex inter-gene dependencies. We utilize a combination of class scatter metrics and heuristic search algorithms to determine all those minimal combinations of genes that have potential to discriminate between the two leukemia classes. A modified perceptron training algorithm further trains the discriminant gene-sets. This process results in a large number of distinct and accurate classifiers (gene sets). We present an algorithm which is then employed to mine these classifiers to discover 'core' patterns from the compositions of the classifier gene-sets. These cores of genes are potentially very useful for a biologist and may reveal much about inter-gene dependencies and gene functions.

1. INTRODUCTION

DNA microarray technology has enabled researchers to simultaneously monitor the expression levels of thousands of genes. Microarray data, generally presented as matrices of expression levels, represents the transcription levels of genes in cells under different conditions, processes or treatments. This information is invaluable in understanding gene function, inter-gene dependencies and underlying biological processes [5; 6; 7]

Recently there has been a lot of interest in classifying various tumors using gene expression data [6; 8; 1]. It has been shown that gene expression data acquired from leukemia patients can be used to build predictors that can discriminate between two acute leukemia subtypes, acute lymphoblastic leukemia (ALL), and acute myeloid leukemia (AML) [1]. To maximize the efficacy of cancer treatment while at the same time reducing its toxicity, it is imperative to target specific therapies to pathogenetically distinct tumor types. Thus, improved cancer classification is invaluable to advances in cancer treatment. Subtypes of acute leukemia, ALL and AML, have similar histopathological appearance. Correct prediction of the subtype of cancer at an early stage from gene expression data can vastly improve the accuracy of diagnosis and effectiveness of treatment. Currently, no single test is sufficient to make a diagnosis - leukemia classification still remains imperfect. Based on gene expression data collected from 72 patients suffering from either ALL or AML, it has been shown [1] that a large number of genes (approximately 1100) have higher correlations with the AML/ALL distinction than can be expected by chance. Classifiers built with small subsets of these genes, selected based upon their individual correlations with the cancer subtypes, have been used to predict the leukemia subtype with some accuracy. Since there is limited knowledge about the functional relevance of most genes, and a strong possibility exists of noise and biasing by inter-gene dependencies and gene interactions, the selection of a set of genes that constitute a good and biologically meaningful classifier is very important. What genes should be chosen to form a classifier and how many genes should be included are two important issues in designing a classifier. Answers to these questions must take into consideration computational tractability, robustness against noise, inter-gene dependencies, and dimensionality of the dataset.

1.1 Exploratory Analysis of Data

Most statistical pattern recognition methods require an abundance of data in order to generate inferences with sufficient confidence. Typically, a dataset on which statistical methods would be relevant consists of a large number (upwards of 1000s) of data points and very few variables (order of 10s) being measured for each data point. In the case of gene expression datasets, each data point consists of a few thousand measured variables (genes) and a few (order of 10s) data points. An exploratory analysis of such a dataset requires that we generate a large number of plausible hypotheses (good classifiers, in our case here) and examine them. Any inference of high confidence can be based only on its consistency with a large fraction of plausible hypotheses. This approach is adopted by us in the methodology outlined below.

1.2 Set of Plausible Classifiers

The complete space of possible *classifiers*, each member of which discriminates ALLs from AMLs, is formed by all possible subsets of the observed genes (which number 6000+), and a weight vector associated with each subset. This space of classifiers is intractably large. Some traditional approaches are based upon the identification of *eigenvectors* of the covariance matrix. The directions of the eigenvectors are not necessarily the optimal discriminators for the classes, and these directions in the sample space are even less relevant given that they represent about 38 samples in a 6000+ dimensional space. (The ALL/AML training dataset consists of only 38 cases [1])

We adopt a strategy that works in the following two phases. During the first phase we use a heuristic search to identify a large number of small-sized subsets of genes that discriminate between classes better than other similar sized subsets. During the second phase we train these classifiers by finding suitable weight vectors to optimize the discrimination potential of each subset. From a relatively large population of such trained classifiers we select a subset of the better performing individual classifiers.

Using the above methodology we demonstrate for the ALL-AML case, existence of gene sets that have 100% accuracy in predicting the subtype of leukemia. We also developed and implemented a perceptron-like algorithm to make the predictions based on the information provided by these gene sets, and a visualization tool to help understand the classifications.

One of the main thrusts behind our strategy is our belief that it is not enough to look at the correlation between individual genes and the classes, but the effect of a set of genes as a whole, given the possibility of inter-gene dependencies.

1.3 Cores: Frequent Patterns in Classifiers

To glean insight from a large set of well-performing classifiers, we introduce an algorithm to *mine* or extract *cores* of genes from the set of classifiers. We define a *core* as a subset of genes that is an integral part of several distinct good classifiers. The strength of a core is defined by the number of candidate classifiers in which the core appears. The presence of many such different cores across many different classifiers can indicate that there may be several different processes leading to the ALL/AML distinction. A core itself is an indication that the genes constituting it are strongly related, at least in the context of the processes leading to the ALL/AML distinction. The examination of cores 'mined' by our algorithm can lead to a better understanding of the workings of the genome with respect to tumors.

1.4 The Leukemia Dataset

The leukemia dataset we have used for our tests is described in [1]. In this dataset there are 72 samples obtained from acute leukemia patients at the time of diagnosis. The training set consists of 38 bone marrow samples (27 ALL, 11 AML) and the independent test set consists of 34 samples (20 ALL, 14 AML). The samples were selected randomly based on availability and are from bone marrow, peripheral blood, childhood and adult cases, etc. Each sample consists of expression level data generated for 6817 human genes by microarrays from Affymetrix. Each numerical value is accompanied by a call value generated by Affymetrix using a proprietary computation.[1]

2. METHODOLOGY AND ALGORITHMS

2.1 Feature Selection Strategies

Feature selection or classifier design is a well studied problem [2; 4]. The goal is to reduce dimensionality, increase computational efficiency and increase accuracy of classification. We define a k-dimensional *feature-set* as a subset of k genes, $F = \{g_0, g_1, g_2, ..., g_k\}$ selected from those n genes $g_0..g_n$ for which expression data is available. A *classifier* G is a feature-set combined with a corresponding set of weights $\{w_0, .., w_k\}$ that helps discriminate between two (or more) classes. Every feature set of size k is associated with a vector $E = \{e_0, e_1, e_2, ..., e_k\}$ of expression levels observed in a sample. there are 72 expression vectors representing 72 different samples for each and every feature set in the leukemia database. A sample represented by the expression levels E corresponding to the genes in the feature set F can be regarded as a point in a k-dimensional space.

There are two problems associated with feature selection on gene-expression data. First, it is difficult to screen features based on their functions, since the functionalities of most genes in biological processes are either unknown or only partially known. Second, it can be misleading to evaluate features for inclusion in a classifier based on individual merit, since genes usually work in groups to regulate biological processes, and since the correlation between any two genes may vary from one process to another. Thus, the features of interest are most likely small sets of highly inter-dependent genes.

Feature selection algorithms are typically sequential (hill-climbing), exponential or randomized. Exponential algorithms (branch and bound, exhaustive) have complexity of O (2^k) where k is the number of features, and are computationally prohibitive for large feature sets. Sequential algorithms have polynomial complexity (O (k^2)) and are widely used. The general strategy is to add or subtract features and follow a hill-climbing approach. However, in a sequential search, the optimal solution is not guaranteed.

Sequential feature selection algorithms are either forward sequential selection (FSS) or backward sequential selection (BSS) [2; 4]. FSS starts with all unique feature sets consisting of just one feature each, and adds one feature at a time to each set. Based on a performance measure, the best feature set is chosen and during the next iteration, new feature sets are generated by adding different single features to the chosen set. This expansion of feature sets continues until there is no improvement in performance. BSS, on the other hand, starts with a single set consisting of all the features and at every stage, new feature sets are generated by subtracting a single feature from the set. The best performing set is now chosen for the next series of subtractions. BSS is held to be superior to FSS for the reason that at any stage, every feature is evaluated in the context of all other features. FSS does not provide this contextual information.

2.2 Search Algorithm

Our algorithm is based on the A^* heuristic search algorithm and it searches the space of all subsets of the complete feature (gene) set. Using a heuristic measure, the algorithm prunes badly performing branches and retains good subsets, as the size of the subsets is slowly reduced. This algorithm is of polynomial time complexity. The heuristic function evaluates the performance of a feature subset - it measures the class-seperation strength of the feature set and is critical to the performance of the algorithm.

An outline of the feature selection algorithm is as follows:

1. Initialize the starting level feature-set G with all genes

[1] There is some controversy regarding these call values, and we have chosen to ignore these call values.

in the sample space S.

2. Loop till number of genes per feature-set has been truncated to the cut-off size C.

3. For each feature-set G with N genes in the previous level, generate N subsets $G_1 ... G_N$ with $(N-1)$ genes each, by shaving off one gene at a time.

4. For each feature-set $G_1 ... G_N$ compute the performance metric J (see next section) using the evaluation function.

5. Save the top T and bottom B feature-sets according to the J criteria for the next level.

6. Update T and B for next level.

7. End Loop [2-6].

T and B are numbers that determine how many feature set nodes from the search tree are to be expanded. T denotes the number of top performing feature sets and B denotes the number of those with bad performance. These are given as parameters to the program, but are modified during program execution. Both T and B are increased gradually as the number of feature sets increase. T and B are coded as absolute numbers of feature sets to expand, steadily increasing with the depth level of the search tree.

The rationale behind choosing the top T and bottom B sets is as follows. Consider the BSS algorithm where only the best node is expanded at every stage. This is done because there is a good probability that the optimal solution will be along the branch whose parent node's performance is best. If we increase the number of nodes that are expanded, we increase the chances of finding the optimal solution. Since the better performing the node, the better the chances of finding the optimal solution along that route, we choose the top T nodes. The gain due to increasing T follows a diminishing returns curve, so we should select a T that is a good tradeoff between efficiency and thoroughness. Furthermore, we are interested in finding a set of good classifiers, and not just one single classifier. It is helpful to have a set of classifiers for the following reasons. One may find that in examining a new sample, data for one or more genes (that are in the classifier) are corrupt or missing, in which case we need to use a classifier for whose genes the data is intact. Also, data may not be available due to practical or clinical reasons. Since one of our objectives of feature selection is to examine patterns in the way classifiers are composed and thereby derive biological insights, it is helpful to have a set of feature sets that perform well. From these feature sets, we can extract groups of genes that are present across feature sets and speculate about their interdependence. On the other hand, by looking at the bottom B classifiers, we not only ensure the persistence of feature sets that may look bad at a higher dimensional stage, and yet might hold promise upon the shaving off of noisy genes, but also provide a useful set of classifiers to compare and contrast against the better performing set while doing the composition analysis.

2.3 Heuristic Evaluation of Classifiers

The classifier evaluation function takes a feature set as input and outputs a quantitative performance measure for that feature set. The metric is the projected strength of a classifier constructed from that feature-set in distinguishing between the classes. Many metrics were considered for this purpose, and their merits were weighed against computational efficiency.

We found that two good class-separability metrics that are computationally friendly are the scatter matrices: S_b (*Between-class scatter matrix*) and S_w (*Within-class scatter matrix*). These $k \times k$ matrices reflect the way sample points are clustered in a k-dimensional feature space (where k is the number of genes constituting a classifier). The scalar metric $trace(S_b)$ is a measure of the average distance of the means of each individual class from the respective global value. It takes large values when the classes (centroids) are well separated, and small values when they are close together. On the other hand, $trace(S_w)$ is a measure of the average, over all classes, variance of features. It takes small values when the classes are close together, and large values when they are spread out. They are calculated as follows:

$$S_w = \sum_{i=1}^{M} P_i S_i$$

where M is the number of classes (2 in our case), P_i is the *a priori* probability of class ω_i and S_i is the covariance of the class ω_i.

$$S_i = E[(x - \mu_i)(x - \mu_i)^T]$$

where x is the expression vector and μ_i is the mean of all feature vectors in class ω_i.

$$S_b = \sum_{i=1}^{M} P_i (\mu_i - \mu_0)(\mu_i - \mu_0)^T$$

where μ_0 is the global mean (over all classes) feature vector

$$\mu_0 = \sum_{i}^{M} P_i \mu_i$$

For the classifier evaluation function we use the metric

$$J = \frac{trace(S_b)}{trace(S_w)}$$

J gives high merit to those clusters that are compact within class, and at the same time are well separated between classes. This is the expectation from an ideal classifier. Thus, by evaluating the branches of the search tree and choosing those that have a higher J value for expansion, we automatically ensure that the search space is largely reduced to include mainly those combinations of genes that are potentially good classifiers.

2.4 Training of Linear Classifiers

Each feature-set, identified by the search as a potentially good class-predictor, is ultimately a linear classifier. The weights associated with the feature-set need to be trained to orient the hyper-plane of separation in k-dimensional space to best classify the samples. This is done using a pocket variant of the perceptron algorithm, which is especially suited for linearly separable classes, suggested in [3]. Unlike the

original perceptron training algorithm which does not guarantee convergence for classes that are not linearly separable, the pocket algorithm always converges to the optimal solution, that is, the one that produces the least number of misclassifications [3]. The unmodified perceptron has been tried in [8] without any consideration for prior feature-selection, and its performance was found to be disappointing. The excellent results we achieve using the perceptron is a validation of our prioritized feature-selection approach.
An outline of this algorithm is as follows:

1. Associate a k-dimensional vector X with the k-dimensional classifier G (where k is the number of genes in the feature-set G) for every sample. X consists of the expression values of the sample $(e_1, e_2, ..., e_k)$ corresponding to each gene $(g_1, g_2, ..., g_k)$ in G.

2. Initialize a weight vector W_0 randomly. Define a stored (pocket) weight vector W_s. Set a history counter H_s of the W_s to 0.

3. Begin Loop.

4. At the i^{th} iteration step compute the update W_{t+1}, according to the perceptron rule: $W_{t+1} = W_t$ - $\rho * Error(X)$
   where ρ is the learning rate, and $Error(X)$ is the summation of all the misclassified vectors X. A vector X is misclassified if it satisfies the following criteria:
   $W' \cdot X > 0$ and X belongs to Class 2 (AML) (or)
   $W' \cdot X < 0$ and X belongs to Class 1 (ALL)

5. Use the updated weight vector to test the number H of training vectors that have been correctly classified. If $H > H_s$ then replace W_s with W_{t+1} and H_s with H, else exit the loop.

6. End Loop.

This algorithm is used to train classifiers G_i from feature-sets F_i by finding corresponding weight vectors W_i. Prediction of the cancer class of an unknown sample m is based on the following rule:
If X_m is the k-dimensional vector of expression values associated with the unknown sample m, and X_m corresponds with G_i in gene composition, then
If $W_i' \cdot X_m > 0$ then m is an AML sample, otherwise m is an ALL sample.

2.5 Mining Cores from Classifiers

Once we have a set of distinct classifiers that can discriminate between the two classes, it would be useful to analyze the compositions of these classifiers to gain deeper insights into the roles of various subsets of genes. Specifically, we would like to see if there are any patterns in the composition of the classifiers, and whether some features frequently come in groups (indicating strong gene-dependencies). We have developed an $O(n^2)$ algorithm to mine such cores from any set of n (of the order of thousands in our current tests) classifiers produced by our feature selection algorithm. An outline of this algorithm is presented below:
Let $SMALLEST$ and $LARGEST$ be the desired cardinalities of the smallest and largest core sets of interest.

1. Initialize an array $G(1...n)$ of n classifiers of different sizes

2. Do (loop)

3. Initialize an empty array of sets, P, and set the number of cores mined, c to 0

4. Initialize an array $intersect_flag(n) to false$

5. for $i = 1$ to $n - 1$, for $j = i + 1$ to n

6. let $R = G(i) \cap G(j)$

7. if $SMALLEST < size(R) < LARGEST$ and $R \notin P$ then $P(c) = R; c = c + 1; flag(i) = flag(j) = true$

8. next j, next i

9. Let $P() \subset Cores$; $G() \leftarrow P(); n = size(G())$

10. if $intersect_flag(x) = true$ for some x then $G(x) \in Cores$

11. While $c <> 0$

The output of the algorithm is the array $Cores$ which contains all the minimal sets of genes that are common to many classifiers. It may be useful to remove all those cores which appear in less than a predetermined number of classifiers. The strength of each core is determined by counting the number of its occurrences in the classifier list. This is an exploratory mining technique that can potentially be very useful to domain specialists looking for information that on gene function, and gene interaction.

3. TEST RESULTS AND DISCUSSION

We have two goals for our methodology. The first is the generation of a set of small sized feature-sets that would work well in classifying the samples. The second goal is to obtain meaningful insight from the potentially large set of classifiers themselves, by discovering patterns in the sets of genes that compose these classifiers.
While seeking the first goal, we found many classifiers with excellent discrimination performance, including classifiers that separate the two classes without a single error in the test and the training sets. The discovery of such a large number of distinct classifiers is an effect of factors such as insufficient data, and complex inter-gene dependences etc. Only some genes in each classifier may be truly responsible for the separation and some others may be present due to spurious statistical correlations. Discovery of core sets of genes can help us in identifying the important genes that contribute to the separation but by themselves do not have enough statistical support to demonstrate the extent of their influence. This is achieved by our second stated goal. We discovered many core sets of genes within the classifiers giving rise to valuable information for a biological scientist.

3.1 Experiments

Our general methodology of experimentation was as follows. We first perform a coarse selection of the genes most likely to be associated with the ALL/AML distinction. This is done to weed out genes that are totally irrelevant or those that do not have a significant correlation with the classes. This step is necessary to bring down the number of genes in the feature space from a few thousands to a more manageable figure such as a few hundreds or fewer. However, note that this step is not essential, and might be counterproductive since it does not consider contextual information. If one has enough computer memory and time constraints are not important, the feature selection algorithm can be applied to the entire feature space. The coarse selection is done by assigning a performance measure FDR (Fisher's Discriminant Ratio) to each gene (without any context) and picking the top N genes. The value of N one chooses depends on how powerful the computer is, and how exhaustive one wants the search to be. Also, prior knowledge of the importance of genes (or suspicion about a gene's importance) can be used to choose genes that might not be automatically selected using this method. The performance measure FDR is actually an approximation of the measure $J = \frac{trace(S_b)}{trace(S_w)}$ for a single dimensional feature set:

$$FDR = \frac{(\mu_1 - \mu_2)^2}{(\sigma_1^2 + \sigma_2^2)}$$

Intuitively, this gives high values for classes whose means are well separated and whose standard deviations are small. FDR assumes that the classes are equiprobable. We then run the feature selection algorithm on the reduced feature space. Choosing the parameters T and B that determine the number of best performing and worst performing feature set nodes to expand at every level depends on the computational resources available.

We ran tests using three different performance metrics for the feature selection algorithm: $trace(S_b)$, J and another metric

$$J_s = \frac{(S_b + (a * S_b * S_w))}{S_w}$$

where a is a tunable variable. J_s biases the J measure to give more importance to S_b and reduces to J when $a = 0$. Our results showed that J almost always outperforms pure S_b. For smaller feature-spaces ($k < 15$), J_s outperformed J for certain values of a. However, J_s is highly dependent on the value of a and the feature space itself. If one can find a good way of tuning a, it might be a useful measure.

3.2 Feature Sets Output by Search

The feature selection algorithm returns a list of all the top T and bottom B classifiers at every stage of the selection process and their associated performance figure. The following is a collage of snipped portions from the output of the feature selection algorithm. Each line represents a classifier - the first field is the number of genes/features constituting the classifier, the second is the classifier itself. (Each gene is denoted by a number in the set. For instance, gene 0 is Zyxin.) The third field is the performance measure for that classifier (in this case, it is S_b.) The *level* is related to the depth of the search tree, and is the same as the number of genes in each feature set.

```
Level # : 6
No of Subsets: 4472

6 {1,2,3,6,9,19} 0.466252
6 {1,2,3,6,9,16} 0.471257
6 {0,5,10,13,14,20} 3.358114
6 {0,13,14,15,20,23} 3.577219
6 {0,13,14,15,20,22} 3.581989

Level # : 7
No of Subsets: 4264

7 {1,2,3,6,9,16,19} 0.581111
7 {1,2,3,6,16,19,24} 0.612764
7 {0,8,13,14,15,20,22} 3.766830
7 {0,10,13,14,20,22,23} 3.841800
7 {0,13,14,15,20,22,23} 3.932278

Level # : 8
No of Subsets: 4066

8 {1,2,3,6,9,16,19,24} 0.706293
8 {0,10,13,14,15,20,22,23} 4.270266

Level # : 9
No of Subsets: 3877

9 {1,2,3,6,9,16,19,21,24} 0.839363
9 {0,10,13,14,15,18,20,22,23} 4.524065

Level # : 10
No of Subsets: 3697

10 {1,2,3,6,9,12,16,19,21,24} 0.978012
10 {0,5,10,13,14,15,18,20,22,23} 4.745728

Level # : 11
No of Subsets: 3526

11 {1,2,3,6,9,11,12,16,19,21,24} 1.143048
11 {0,5,8,10,13,14,15,18,20,22,23} 4.930569

Level # : 12
No of Subsets: 3363

12 {1,2,3,4,6,9,11,12,16,19,21,24} 1.310686
12 {0,5,7,8,10,13,14,15,18,20,22,23} 5.102854

Level # : 13
No of Subsets: 3208

13 {1,2,3,4,6,9,11,12,16,17,19,21,24} 1.480636
13 {1,2,3,5,6,8,9,12,16,17,19,21,24} 1.554465
13 {0,5,8,10,12,13,14,15,18,19,20,22,23} 5.179070
13 {0,5,7,8,10,13,14,15,17,18,20,22,23} 5.272804

Level # : 14
No of Subsets: 3060

14 {1,2,3,4,6,7,9,11,12,16,17,19,21,24} 1.652921
14 {2,3,4,5,6,7,8,9,11,12,16,19,21,24} 1.805969
14 {0,4,5,7,8,10,13,14,15,17,18,20,22,23} 5.440442

Level # : 15
No of Subsets: 2919
```

3.3 Trained Linear Classifiers

From the output of the feature selection program, we either choose only those interesting feature-sets by using a performance threshold, or all of them at once, and run the *batch−prediction* program on the chosen sets. This program constructs the corresponding classifier from each feature-set in the batch. It works by training the weights associated with the feature-set (using the modified-perceptron algorithm) on the training data. It then tries to predict the tumor classes for all the samples in the independent and the training sets using the classification rule. Thus, the output of this program is the final set of classifiers extracted from the feature-space.

The following is a small sample of the output from the *batch − prediction* program. The first field is the number of genes in the classifier, the second is the classifier itself, and the third field is the number of mis-classifications over both the training and the independent datasets.

```
8 {0,5,10,14,15,18,22,23} 0
8 {0,5,10,13,14,15,20,22} 0
10 {0,5,8,13,14,15,18,20,22,23} 0
10 {0,8,10,12,13,14,15,20,22,23} 2
10 {0,3,10,13,14,15,18,20,22,23} 1
12 {0,7,8,10,11,13,14,15,18,20,22,23} 2
12 {0,4,5,10,13,14,15,18,20,21,22,23} 1
12 {0,4,8,10,13,14,15,17,18,20,22,23} 3
13 {0,4,5,8,10,12,13,14,15,18,20,22,23} 1
13 {0,5,8,10,12,13,14,15,17,18,20,22,23} 0
13 {0,4,5,7,10,13,14,15,17,18,20,22,23} 2
13 {0,4,5,8,10,13,14,15,17,18,20,22,23} 1
19 {0,4,5,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23} 0
19 {0,4,5,7,8,9,10,11,12,13,14,15,16,18,19,20,21,22,23} 1
22 {0,1,3,4,5,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23} 1
23 {0,1,2,3,4,5,6,7,8,9,10,12,13,14,15,16,17,18,19,20,21,22,23} 0
23 {0,1,2,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23} 2
23 {0,1,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23} 2
```

The actual output of good classifiers runs into hundreds. From a starting feature-space of 60 genes, we found between 100-150 classifiers with 100% accuracy of prediction, while varying T between 500 and 1000. There were also a large number of classifiers with just 1 or 2 misclassifications.

The composition of an example predictor, which accurately classified all the 72 samples is as follows:

Zyxin, KIAA0097 gene, KIAA0212 gene, Macmarcks, Heat Shock Protein 70 Kda(Gb:Y00371), Induced Myeloid Leukemia Cell Differentiation Protein MCL1, FTL Ferrtin light polypeptide, ADA Adenosine deaminase, LYN V-yes-1 Yamaguchi sarcoma viral related oncogene homolog, CD33 antigen (differentiation antigen), Oncoprotein 18 (Op18) gene, CTSD Cathepsin D (lysosomal aspartyl protease), Major histocompatibility complex enbhancer-binding protein MAD3, Transcription Factor RELB.

The weight vector W associated with its prediction is (-19531,2287,20384,33396,16255,-8092, -7995,43902,-9986,-6504,26279,-32220,1496,-5463)

Most predictors seemed to contain the gene Zyxin. The composition of an example core (that did not contain Zyxin) found in over 44 different classifiers of different sizes is as follows:

Macmarcks, CD33 CD33 antigen (differentiation antigen), FAH Fumarylacetoacetate, CTSD Cathepsin D (lysosomal aspartyl protease), CCND3 Cyclin D3, Phosphotyrosine independent ligand p62 for the Lck SH2 domain mRNA, Leukotriene C4 synthase (LTC4S) gene, RETINOBLASTOMA BINDING PROTEIN P48.

3.4 Extracted Gene-Cores

Once a large number of such good classifiers are generated and their performance is evaluated, our next task is to look for information within the structural composition of these classifiers. We would like to understand the biological significance of why and how certain classifiers can discriminate between the two classes accurately while others cannot. We are looking for answers to questions such as: Is there any significance in the way these genes are combined? What is the meaning of strong patterns within the classifier compositions? Can we look at two classifiers that are similar in all respects except for a few and draw conclusions from their relative performance? Can we speculate about the function of an unknown gene function given the knowledge of the functions of other known genes within the classifier?

An approach to answering these questions is to look for patterns of similarity within the classifier compositions. We use the $core-extraction$ algorithm to reveal the various combinations of genes that are common to multiple distinct classifiers. We used various parameters for the core size and extracted a large number of cores, i.e gene subsets that occur in many different classifiers. Most cores are found to be classifiers by themselves. It is interesting to note how these cores can be combined with certain subsets of genes to produce good classifiers but lose in predictive power when combined with other gene subsets. Domain knowledge of their functions can be helpful in understanding the nature of the processes that regulate the development of the tumors themselves.

Some sample snipped output from the core extraction algorithm is given below. The first field represents the number of genes in the core, the second is the core itself and the third field represents the number of accurate classifiers in which the core occurs.

```
5 {5,15,18,23,30} 36
5 {5,10,15,18,30} 15
5 {0,4,10,13,17} 12
6 {2,5,9,15,16,18} 8
6 {0,5,15,18,23,29} 26
6 {15,16,18,23,30,31} 6
6 {0,2,15,18,23,29} 5
6 {2,7,16,18,23,30} 7
8 {5,15,17,18,23,29,30,39} 11
```

The following observations and conclusions are made based on the number and composition of the best and worst classifiers returned by the feature-selection algorithms:

1. The average classifier performance is best when the number of genes constituting the classifier is below 25. From this information and the fact that the number of genes having significant individual correlation with the ALL/AML distinction is two orders of magnitude greater, we can conclude that the interactions/mutual dependencies play an important role in cancer differentiation and that the larger the subset of genes, the greater the possibility of errors due to noise. This is a good argument for a metric that evaluates features in the context of other features (i.e. the classifier as a whole) and not purely individual correlations.

2. There are a huge number of classifiers with zero prediction errors. Their compositions vary quite a bit, indicating that there are many distinct hyper-planes of separation between the two classes. Using multiple classifiers in the prediction of a new sample can be helpful as a corroborative aid. If many distinct classifiers confirm the prediction, then it is a stronger and better prediction.

Figure 1

3.5 Visualization

We developed a graphical tool to visualize the separation of the classes using a given feature-set/classifier. This tool graphically displays in 2 dimensions the Euclidean distances of each sample from the centroids of the sample vectors. Each sample can be considered an k dimensional vector of expression values for k genes. Each sample on the 2D graph is separated from the class centroids by the same Euclidean distance as in k dimensional space. Thus we can see the non-weighted separation between the two classes in 2 dimensions for any given classifier set. Figures 1 and 2 show the outputs produced by the visualization of two classifiers A and B, each consisting of 15 genes. The only difference between the two is that one of the classifiers has the gene MAD3 and the other has MCL1. Comparison of the two figures (for instance, the pronounced effect of this change on the samples 12 and 35) by a biologist may provide some insight into the roles of these genes in the ALL/AML distinction.

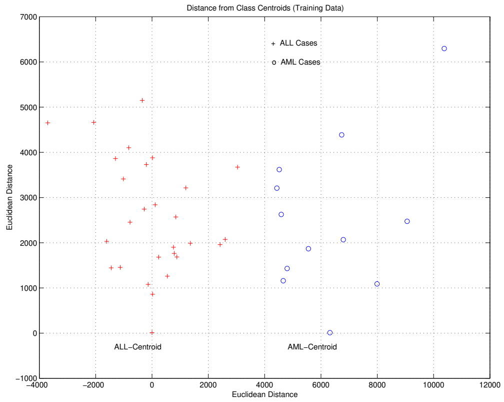

Figure 2

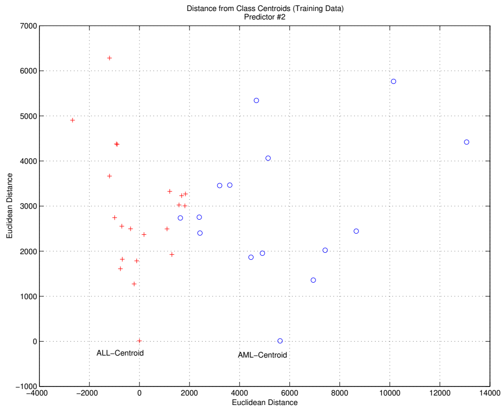

Figure 3

4. CONCLUSIONS

Statistical methods of classification are inadequate for the cancer classification problem mentioned above since there exist many compositionally distinct classifiers that accurately discriminate between ALL and AML. One factor responsible for abundance of distinct and well performing classifiers is the insufficiency of observed data for 6000+ dimensional feature space. This also demonstrates that the best discriminant surface for such a dataset may not be found along the directions of principal variance. Our heuristic search based method of selecting features generates a large number of distinct feature-sets that are in turn used to build accurate classifiers using a modified perceptron training based algorithm. The twin problems of computational intractability and gene inter-dependencies are alleviated by our approach of using a heuristic for expanding only a small set of feature-set nodes based on the performance of each feature-set. Further the scatter based metrics we use are easily computed and compare well with the output of the perceptron for class separability information. Our efficient algorithm for extraction of core gene-sets from the classifiers reaped by the search, produces many interesting gene-cores whose analysis can be very valuable and revealing to biologists.

5. REFERENCES

[1] T.R.Golub, D.K.Slonim, P.Tamayo, C.Huard, M.Gaseenbeek, 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(5349)*, pages 531–537, 1999.

[2] D.W.Aha and R.L.Bankert. A comparative evaluation of sequential feature selection algorithms. In *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 1–7, 1995.

[3] S.I.Gallant. Perceptron based learning algorithms. *IEEE Transactions on Neural Networks*, 1(2):179–191, 1990.

[4] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition*. Academic Press, San Diego, CA, 1999

[5] N.S.Holter, M.Mitra, A.Maritan, et al. Fundamental patterns underlying gene expression profiles: Simplicity from Complexity. *Proceedings of National Academy of Science*, V.97, no. 15, pages 8409-8414, July 18, 2000

[6] A.A.Alizadeh, M.B.Eisen, R.E.Davis, C.Ma et al. Different types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature, 403:503-511, 2000*

[7] M.B.Eisen, P.T.Spellman, P.O.Brown, D.Botstein. Cluster analysis and display of genome-wide expression patterns *Proc. Natl. Acad. Sci., 95:14863:14868, 1998*

[8] S.Dutoit, J.Fridlyand, T.P.Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. *Technical report 576*, Dept. of Statistics, UC Berkeley.