Evolutionary Algorithms for Finding Optimal Gene Sets in Microarray Prediction

J.M.Deutsch
University of California, Santa Cruz, U.S.A.

Abstract

Motivation: Microarray data has been shown recently to be efficacious in distinguishing closely related cell types that often appear in different forms of cancer, but is not yet practical clinically. However the data might be used to construct a minimal set of marker genes that could then be used clinically by making antibody assays to diagnose a specific type of cancer. Here a replication algorithm is used for this purpose. It evolves an ensemble of predictors, all using different combinations of genes to generate a set of optimal predictors.

Results: We apply this method to the leukemia data of the Whitehead/MIT group that attempts to differentially diagnose two kinds of leukemia, and also to data of Khan et. al. to distinguish four different kinds of childhood cancers. In the latter case we were able to reduce the number of genes needed from 96 to less than 15, while at the same time being able to classify all of their test data perfectly. We also apply this method to two other cases, Diffuse large B-cell lymphoma data(Shipp et al., 2002), and data of Ramaswamy et. al. on multiclass diagnosis of 14 common tumor types.

Availability: http://stravinsky.ucsc.edu/josh/gesses/

Contact: josh@physics.ucsc.edu
Introduction

cDNA and oligonucleotide microarrays have been used with great success to distinguish cell types from each other, and hence has promising applications to cancer diagnosis. While the histopathology of two cells may appear very similar, their clinical behavior, such as their response to drugs can be drastically different. The use of microarrays has been shown in many cases to provide clear differential diagnosis rivaling or surpassing other methods and leads to a clustering of data into different forms of a disease(DeRisi et al., 1996; Alon et al., 1999; Perou et al., 1999; Zhu et al., 1998; Wang et al., 1999; Schummer et al., 1999; Zhang et al., 1997; Alizadeh et al., 2000; Golub et al., 1999; Khan et al., 2001).

Many approaches have been used to classify microarray data. These include the use of artificial neural networks(Khan et al., 2001; Furey et al., 2000), logistic regression(Li & Yang, 2001), support vector machines(Brown et al., 2000; Furey et al., 2000), coupled two-way clustering(Getz et al., 2000), weighted votes - neighborhood analysis(Golub et al., 1999) and feature selection techniques(Xing et al., 2001). For much of the data all these techniques appear to give similar results and their performance improves as the amount and quality of data increases. For example work on the classification of two different leukemias(Golub et al., 1999) attempts to classify 34 test samples based on 38 training samples. On prediction of test data, different predictors make anywhere from 0 to 5 mistakes. On the other hand, recent work on small round blue cell tumors (SRBCT) attempted to classify 20 test samples based on 63 training samples(Khan et al., 2001). They were able to classify all test data correctly into one of four separate categories. They were able to do this with a single layer neural network that considered only 96 genes.

To classify samples using microarray data, it is necessary to decide which genes should be included in a predictor. Including too few genes will not discriminate in a detailed enough manner to classify test data correctly. Having too many genes is not optimal either, as many of the genes are largely irrelevant to the diagnosis and mostly have the effect of adding noise, decreasing the “information criterion”(Li & Yang, 2001; Akaike, 1974; KP Burnham, 1998; Schwarz, 1976). This is particularly severe with a noisy data set and few subjects. Therefore an effort is made to choose an optimal set of genes for which to start the training of a predictor. This is done in a variety of different ways, such as a kind of neighborhood analysis(Golub et al., 1999), principal component analysis(Khan et al., 2001), or gene shaving(Hastie et al., 2000) A predictor can then be developed from this carefully chosen subset of genes.

Recent work(Li & Yang, 2001) addressed the problem of gene selection for a leukemia data set(Golub et al., 1999). They initially ranked genes as had been done in the first analysis(Golub et al., 1999) and used the top ranked genes. They varied the number they included and found no clear indication of any optimum number aside from the conclusion that the number should be much smaller than the 50 that had been originally used(Golub et al., 1999).

Here we develop gene selection further by making it an integral part of the prediction algorithm itself. Instead of using all of the highest ranked genes, we find an effective method to greatly reduce this
number. This can be done because gene expression tends to be highly correlated, making many of the initially chosen genes redundant or even deleterious because of the problem of added noise.

The method introduced here is named GESSES (genetic evolution of sub-sets of expressed sequences). It makes use of a kind of evolutionary algorithm known as a replication algorithm that has been extensively used in quantum simulations (Ceperley & Kalos, 1979) and protein folding (Garel & Orland, 1990). It finds a set of highly relevant genes by considering a whole ensemble of predictors, any of which use a different set of genes. As the predictors evolve, more genes are added to each predictor. It eventually produces an ensemble of predictors each of which can be tried on test data.

In the case of small round blue cell tumors, GESSES reduces the number of genes from 96 down to below 15 while still predicting the test data perfectly. Some of the perfect predictors have only ten genes.

It is hoped that GESSES will have applications in the clinical diagnosis of cancer (He & Friend, 2001). At the moment, microarray experiments are too costly and time consuming to be used clinically. However if a subset of marker genes could be found whose expression levels could then be obtained using relatively inexpensive antibody assays, this might become a practical method. Therefore for this purpose it is important to use as few genes as possible and still obtain an accurate diagnosis of the disease.

With the same algorithms applied to leukemia data of Golub et. al., we find conclusions in accord with Li and Yang (Li & Yang, 2001) that there is no clear indication of an optimum number of genes to use in a predictor. We find a range of predictors some that predict the test data perfectly but many predictors that get several samples wrong. This is also in accord with other groups work (Golub et al., 1999; Furey et al., 2000). Without further data or further biological information, it is probably not possible to do better than this.

GESSES was applied to several additional data sets, two data sets related to Diffuse large B-cell lymphoma (DLBCL)(Shipp et al., 2002) and work (Ramaswamy et al., 2001) on the diagnosis of 14 different classes of tumors using microarrays. In these cases, GESSES was able to reduce the number of genes needed to make a prediction of a given error rate. In particular, the work of Shipp et. al. evaluated their predictors using leave one-out cross validation. They first compared DLBCL data with follicular lymphoma (FL). With the same methodology for evaluation, GESSES outperforms their 30 gene predictor by correctly classifying all samples, instead of 91%, and only used predictors ranging from 4 to 12 genes. With Shipp et. al’s second data set, concerned with the prediction of patient survival of DLBCL under chemotherapy, GESSES fairs well as compared with the original method used. It should be noted however that the use of leave one-out cross validation as a measure of predictive power is not optimal, but is necessary when the amount of data makes division into training and independent test data problematic. To get a better estimate of predictive value, the data on DLBCL versus FL was divided into training and test data. The results on the test data were highly statistically significant.

This paper is organized as follows. We discuss the algorithm used by first providing an overview of its basic features and then in detail by first defining the terminology and concepts used. Then we discuss the predictor used, the scoring function, the kind of evolutionary algorithms used and the annealing
schedules. We then apply this to the SRBCT, leukemia data, two DLBCL data sets, and multiclass tumor data. After this, we report on a variety of tests that were used to validate the method and results. Last, we make some concluding remarks.

**The Algorithm**

**Overview**

The algorithm can be divided up into several parts. First we are interested in the evolution of an *ensemble*, or population, of predictors. What distinguishes one predictor from another is the subset of genes it utilizes in making a prediction.

The most successful predictors will be the ones making fewest mistakes on test data. To determine which predictors are most successful, we utilize a scoring function which gives higher scores when more data points are correctly classified, that is the smallest classification error. Because we can only use a fixed amount of training data when evolving the predictors, we use leave-one-out cross validation to calculate the score for a certain predictor. We obtain better predictors by adding an additional term to the scoring function to give higher marks to predictors that do a good job of grouping the data into well separated clusters, each cluster corresponding to the same type of cancer.

We would like a method that searches through a large number of different subsets of genes to come up with a population of the highest scoring predictors. This is often referred to as a *wrapper* method and has been the subject of a large body of research (Langley, 1994; Kohavi & John, 1997).

Most genes have little or no predictive value. The more of them that are included as possible choices, the more noise is added to the predictions which leads to a degradation in the performance of the prediction ensemble. Therefore it is necessary to apply some initial *filter* (Xing *et al.*, 2001) method to construct an initial gene pool containing the most likely candidate genes. We use a simple method of ranking genes similar to previous work (Ben-Dor *et al.*, 2000).

We employ several methods for evolving our population of predictors. We produce offspring by random mutations and deletions of genes, with the number of replications of a particular predictor depending on how the mutations effect the scoring function. The notion of temperature is employed to control the degree to which less favorable mutations are kept in future generations. The higher the temperature, the more unfavorable predictors are kept. We slowly cool the system so that eventually we weed out all but the most fit predictors. This is a kind of simulated annealing.

In addition we employ deterministic methods of evolution that try many combinations, only keeping the ones that score highest.

We will discuss all of these features now in more detail starting with some basic terminology.
Terminology

We have samples of microarray training data \( D_t \equiv \{ D_1, D_2, \ldots \} \) with each sample consisting of \( N \) genes. Corresponding to the \( i \)th gene of a sample is its expression level \( e_i \). The complete set of genes \( G_t \) is the collection of genes 1 through \( N \) and we will consider subsets of \( G_t \), for example the subset \( \alpha_1, \alpha_2, \ldots, \alpha_m \) (e.g. genes 2, 5, and 9), which we denote \( G_{\alpha} \). For this subspace of genes the vector of expression levels \( e_{\alpha} \equiv (e_{\alpha_1}, \ldots, e_{\alpha_m}) \). The number \( m \) of genes in this subset is denoted \( |\alpha| \), which in this case is \( m \).

The set of possible types is denoted \( T \). Each sample \( D \) has a classification of type \( T \), in this case the type of cancer, which can take one of \( |T| \) values.

We introduce the usual definition of the Euclidean distance between samples \( D_a \) and \( D_b \) on the subspace \( G_{\alpha} \):

\[
d^2_{\alpha}(a,b) = \frac{1}{|\alpha|} \| e^a_{\alpha} - e^b_{\alpha} \|^2 = \frac{1}{|\alpha|} \sum_{i=1}^{|\alpha|} (e^a_{\alpha_i} - e^b_{\alpha_i})^2
\]

(1)

where \( e^a_{\alpha} \) and \( e^b_{\alpha} \) are the expression levels of samples \( a \) and \( b \) respectively, for genes \( \alpha_1, \alpha_2, \ldots, \alpha_m \).

Predictor

We define a predictor \( P \) as a function that takes a data sample \( D \) and outputs a type \( T \), in this case the type of cancer that is associated with that data. That is \( P(D) \rightarrow T \).

In this work we will use a k-nearest neighbor search (Duda & Hart, 1973) to construct the predictor. In the results reported below, we use \( k = 1 \), that is, the set of samples that forms the training data \( D_t \) is compared with the target sample \( D \) by finding the distance using eqn. (1) between \( D \) and each vector in the training set. The sample in the training set closest to \( D \) gives the classification \( T \) of \( D \). The distance depends on what subspace of genes \( G \) is used hence the predictor depends both on the training data and \( G \).

We will use variants of this basic predictor when constructing a scoring function that we discuss below. For this we will not only need the closest point, but the values of the distances to all sample points.

Scoring function

The scoring function is used to determine how well the predictor predicts data. By definition we cannot use any information from the independent test data in the development of the predictors. Therefore we consider only the training data \( D_t \) to determine the fitness of a predictor. In other words, we need to score the effectiveness of the predictor using only \( D_t \). This is done as follows.

We consider one point \( D_p \) in \( D_t \) as a pseudo test data point, and eliminate this point from \( D_t \), calling the resultant training data \( D_t - D_p \). We then loop over all points in \( D_p \in D_t \) in the following steps.

1. We find the set of distances between \( D_p \) and points in \( D_t - D_p \).
2. If the type of the point giving the shortest distance matches the type of \( D_p \), we add 1 to the scoring function. Otherwise we add nothing and skip the remaining steps, continuing to loop over the remaining \( D_p \)'s.

3. We consider the distances grouped by the classification type of the target points. We consider the shortest distance of each type from \( D_p \), which we call \( d_1, d_2, \ldots, d_{|T|} \).

4. Of these we take the two shortest, \( d_i \) and \( d_j \) and add \( C|d_i^2 - d_j^2| \) where \( C \) is a constant chosen so that the value of these added terms is \( \ll 1 \).

The first part of the scoring function described in 2 counts the number of correct answers. When the experimental data is predictive, one expects that as the predictors improve, all the training data is correctly classified. However many predictors will be able to predict the training data perfectly but have differing abilities at generalization, that is some are better than others at classifying test data. To optimize prediction of test data, we add a second part to the scoring function, described in 4, which maximizes the separation between the different classes. Of several different methods that were tried, this one worked the best.

The scoring function depends on the predictor, which in turn is determined by the training data and the subspace of genes \( G \). We will denote this latter dependence as \( S_G \).

Initial Gene Pool

Often it is necessary to narrow down the genes that are considered from the many thousand that are measured on the microarray down to of order \( 10^2 \) that are most relevant. There are many ways of doing this, one common method being principle component analysis. For the purposes here we choose instead a different method that is highly effective and similar to one previously used (Ben-Dor et al., 2000).

We consider how genes distinguish two types \( t_1, t_2 \in T \) from each other. For each gene \( g \in G_t \) we consider its expression levels in the training samples. We rank all the training samples in terms of the expression level of \( g \). We are looking for genes that for high levels give type \( t_1 \) and for low levels give type \( t_2 \) (or vice-versa). When ranked this way, they sometimes will perfectly separate, that is the first part of the list is one type, and the last part is the other. These genes are ranked the highest. Most of the time however, a gene will not separate so clearly and there will be overlapping regions. Those with more overlaps of different types are ranked lower. In this way we have a ranking of the genes that are best able to distinguish \( t_1 \) from \( t_2 \), and we pick the top \( M \) genes.

Specifically, we loop over the ranked list of genes, and define the cumulative number of genes of type \( t_1 \) seen up to the \( i \)th iteration to be \( n_1(i) \), and the analogous quantity for type \( t_2 \) to be \( n_2(i) \). Denoting the total number of genes of types \( t_1 \) by \( N_1 \), we define the normalized \( n_1(i) \equiv n_1(i)/N_1 \), and similarly \( n_2(i) \equiv n_2(i)/N_2 \). If the gene ranking perfectly separates for a gene, that means that one of the types, say \( t_1 \) will be only seen initially and so \( n_1(i) \) will become unity before any samples of type \( t_2 \) are looped over. However if they are completely randomly interspersed then on average \( \gamma_1(i) \) and \( \gamma_2(i) \)
will be the same. As a measure of this we consider the point \(i_c\) where \(\gamma_1(i_c) = 1 - \gamma_2(i_c)\). We use \(\gamma_1(i_c)\) as the figure of merit, or \(\gamma_2(i_c)\), whichever one is greater than \(\frac{1}{2}\). For perfect separation this is unity whereas for random data it is \(\frac{1}{2}\).

We then consider all distinct combinations of \(t_1\) and \(t_2\) and pick the best \(M\) genes from each combination. Genes may overlap, narrowing the initial pool. This is our initial set of genes \(G_i\) that we will consider.

**Evolution Algorithms**

Starting off with an ensemble of different gene subspaces we want to determine rules to evolve it to a new one that gives a better set of predictors. To do this, we have to have a measure of how well a predictor classifies samples into separate types. We do this by means of the scoring function described above. The evolutionary methods described below show how to utilize the scores to determine which predictors are kept and which are eliminated.

**Statistical Replication**

In analogy with statistical mechanics, we can think of the scoring function as (negative) energy and invent a dynamics that evolves them towards the highest scoring (lowest energy) states. We do this at finite temperature to allow the system to accept predictors that occasionally may be less fit than their predecessors to get rid of local minima in predictor space and to allow for a diverse population of predictors.

Suppose the system has evolved to an ensemble of \(n\) gene subspaces \(E \equiv \{G_1, G_2, \ldots, G_n\}\), we will now employ a variant of a replication algorithm used in other contexts (Ceperley & Kalos, 1979) to replicate and modify each of the \(G_i\)'s.

1. For each \(G \in E\) we produce a new subspace as follows.
   
   (a) A set of genes \(G\) has genes \(\{g_1, g_2, \ldots, g_m\}\). We randomly mutate genes according to three possibilities:

   i. Add an extra gene: We choose a randomly chosen gene \(g_r\) from the initial set \(G_i\), and add it to \(G\), producing a new set \(G'\) of genes \(\{g_1, g_2, \ldots, g_m, g_r\}\). If \(g_r \in G\), \(G' = G\).

   ii. Delete a gene: We randomly delete a gene from \(G\) producing a new set with \(m - 1\) total genes.

   iii. Keep \(G\) the same.

   (b) We compute the difference in the scoring functions \(\delta S = S_{G'} - S_G\).

   (c) We compute the weight for \(G'\), \(w = \exp(\beta \delta S)\), where \(\beta\) is the inverse “temperature”.

2. Let \(Z\) denote the sum of these weights. We normalize the weights by multiplying them by \(n/Z\).
3. We replicate all subspaces according to their weights. With a weight $w$, the subspace is replicated $\lceil w \rceil$ and an additional time with probability $w - \lfloor w \rfloor$. Here $\lfloor w \rfloor$ denotes the largest integer $< w$.

Mutations as described in 1(a)ii and 1(a)iii can be optionally added, as the algorithm works well with 1(a)i.

In summary, every subspace in the system is mutated and replicated in accordance with how much fitter it was than its predecessor. By carefully normalizing the system, the number of subspaces in the ensemble stays close to $n$. Note that we can also do more than one potential mutation in step 1. We will generalize this to allow $n_m$ potential mutations.

**Annealing**

As the system evolves, the scoring function gives similar answers for all members of the ensemble. In order to improve convergence, it is useful to make the temperature a function of the spread in scores (or energy). A variety of schedules for the temperature were tested. We will focus on two schedules. The first is a relatively simple one for which we have obtained good results. The second is more complex but more efficient and robust.

1. After every iteration, $\beta$ is multiplied by a constant, that is $\beta = I \beta_{old}$. We have worked mostly with $I = 1.2$ though small values would theoretically be more reliable however be less numerically efficient. To further speed up the algorithm we compute the maximum spread in scores between different members of the ensemble, $\Delta E$. If $2/\Delta E$ is larger than $\beta$, we set $\beta$ equal to it instead.

   This is particularly useful because the scoring function has two basic components. The first adds unity every time a sample is correctly classified. The second adds a much smaller number proportional to the constant $C$ defined above, which is chosen to make this second component $\ll 1$. The second component tries to maximize the separation between the different classes. When the predictors have evolved so that they are all classifying the training data correctly, we would like the second part to take effect. By lowering the temperature by a schedule such as the one above, the algorithm will then select for predictors that maximize the second part of the scoring function. This leads to a much better set of genes.

   Note that if we only multiplied $\beta$ by a constant factor at every step, the algorithm would still work, but takes longer to converge since the energy scale suddenly shrinks when all the training data is classified correctly. However it takes many iterations for the temperature to cool to a value comparable with this scale.

2. We use a more typical measure of the spread in energy, the standard deviation of the energies, $\sigma_E$ for different members of the ensemble, instead of the maximum spread as was done above. Instead of picking the initial $\beta$ arbitrarily, we take it to be $3/\sigma_E$. Because the above algorithm can raise $\beta$ too quickly when we replace $\Delta E$ by $\sigma_E$ (which is smaller), we use an algorithm which smoothes out the change of $\beta$. We
do so by considering a geometric mean for the new value of $\beta$,

$$\beta_1 = \sqrt{\beta_{old}(1/\sigma_E)}$$

We also consider $\beta_2 = I \beta_{old}$ which is the same step scheme as used above. Our new value of $\beta$ is the larger of $\beta_1$ and $\beta_2$.

As with the first scheme above, this quickly changes energy scale when all training examples are correctly classified, but cools down slowly enough so as not to get trapped in local minima.

This schedule is also useful because it allows us to define a simple termination condition when all moves (additions and deletions) are allowed. In this case the condition is that the ensemble is unchanged for 10 consecutive iterations. In practice the system terminates fairly rapidly because eventually the temperature decreases to essentially zero, leaving only a small number of systems left in the ensemble.

**Deterministic Evolution**

As an alternative to the statistical replication method above, we also employed a method that is computationally more expensive but that often performs better. The statistical method does not explore all possible combinations of genes at each stage of growth. This can miss optimal gene combinations. We get around this by a deterministic exploration of the optimum gene combinations at every step. A single step goes as follows:

1. Construct all distinct unions of the $G$'s in the ensemble $E$ with individual genes $g_i$ in the initial gene pool $G_i$, i.e. $g_1, g_2, \ldots, g_m, g_i$.

2. Sort all of these combinations by their score, keeping the top $n_{top}$ of them.

To save computer time we tried various values for $n_{top}$. It was found that $n_{top} = n$, (the number of $G$'s in the ensemble) performed quite well. Another variant was to construct only half the unions and keep the top $n$, for computational efficiency.

**Results**

We now show results from the application of the above algorithm to two data sets, work on the small round blue cell tumors (SRBCT) of childhood and work on human acute leukemia (Golub *et al.*, 1999).

**SRBCT Data**

Small round blue cell tumors (SRBCT) of childhood are hard to classify by current clinical techniques. They appear similar under a light microscope and several techniques are normally needed to obtain an accurate diagnosis. The paper (Khan *et al.*, 2001) used microarrays to study their classification using of a single layer neural network. This work differed from previous studies in that they were attempting
to distinguish between four different cancer types instead of the more usual 2. They used 63 samples for training and tested with 20. By using a clever method combining principle component analysis and sensitivity of their neural network to a gene, they were able to reduce the number genes needed to 96 yet still classify all different forms of cancer in test data perfectly.

Here we use the same data set to reduce the number of genes needed and still classify the test data perfectly.

Starting with their data set of 2308 genes, we constructed the initial pool of genes by considering how well a gene discriminates type $i$ cancer from type $j$, as described above. Since there are 4 possible types, we have 6 combinations of $i$ and $j$. For each of these we take the top 10 genes best able to discriminate for each $i, j$ pair. This gives a total of 50 genes, because it turns out that 10 of these overlap between groups. The genes found were significant far beyond what would be expected for random data. For example in comparing EWS with BL training samples, more than sixty genes displayed a perfect separation of expression level.

We then evolve these gene subspaces according to the statistical replication method described above first using only addition of genes as described in the section on statistical replication, 1(a)i. The + points in fig. 1 shows the average number of genes in a predictor as a function of the number of iterations, or generations. The average is over the ensemble of predictors. It starts to level off significantly at the 38th generation, because the addition of further genes does not improve the scoring. The dashed line with × points shows how as a function of iteration the predictors fair with the test data of 20 samples. The vertical axis is the average number of incorrect assignments, again averaged over all predictors in the ensemble. By the 26th generation, more than 90% of predictors perform perfectly with the test data, and by the 41st generation, all predictors perform perfectly using an average of 28 genes.

A useful variant of the above it to use the other two mutational moves referred to in the section on statistical replication, 1(a)ii and 1(a)iii. We started with the same initial pool of genes as above. We used the second cooling schedule described above and evolved the system. The results are shown in fig. 2. The average number of dimensions rises to a maximum of about 16 after 32 iterations, while the number of wrong classifications decreases from about 9 down to about 0.5. By iteration 26 all members of the ensemble classify the training data perfectly. At this point, as was expected, the temperature falls very rapidly, so that the scoring function only probes its small second piece, (see the section on the scoring function, item 4. Now the temperature drops and most classifiers are predicting perfectly. Eventually the systems predict the test data perfectly and we continue to cool it until the termination condition is met described above (see the section annealing). In other words, that the ensemble remains identical for 10 consecutive iterations. For the purposes of these figures, the number of distinct systems is the number of parents that the ensemble has in common. In other words, replications often give rise to multiple copies of a mutated predictor, or no copies at all. The number of distinct systems is the number of independent predictors that do not represent additional copies.

We next use the deterministic evolution method described above only constructing half the unions.
The + point in fig. 3 shows the average number of genes in a predictor as a function of the number of generations. In this case it plateaus off sooner, after about 15 generations. The comparison with the test data is shown by the dashed line with × points. Here all predictors perform perfectly when the average number of genes in a predictor is 15.1. Here we only used half the possible unions and kept \( n_{\text{top}} = 50 \).

Encouraged by the above results we did a larger run starting with an initial pool of 90 genes of which 15 overlapped, giving a total of 75 initial genes. Evolving these with \( n_{\text{top}} = 150 \) gives the results shown in fig. 4. Of the top 100 predictors, all predicted the test data perfectly. The average number of genes in a predictor was 12.7.

With this same initial pool of 75 we also ran the statistical algorithm allowing for all mutational moves, as was done for fig. 2. The results are shown in fig. 5. The features of this curve are similar to those for the run with 50 initial genes, fig. 2. After iteration 28 the test data is predicted perfectly and the temperature rapidly decreases. At iteration 65 all system predict the test data perfectly. The temperature is further decreased until a steady state solution is reached, in this case where the number of distinct systems is 3.

With this data, GESSES can be used to give an ensemble of predictors that have perfect or near perfect performance. However if the initial gene pool is reduced to below 50 distinct genes, it degrades. For example, starting with the top 48 genes (giving 41 distinct genes) with \( n_{\text{top}} = 41 \) leads to a set of predictors that make an average of 0.439 mistakes, and an average number of genes of 11.24. Despite this, one should keep in mind that over half the predictors predict the test data perfectly. But starting with only the top 24 genes (20 distinct genes) with \( n_{\text{top}} = 20 \) leads to a set of predictors that make an average of 1.45 mistakes, and an average number of genes of 10.95.

The implementation of GESSES is quite efficient and the above results took of order a few minutes to complete on a modest 450 MHz Celeron machine, using of order 5 Mbytes RAM.

The genes found by these methods are mostly a subset of those found previously(Khan et al., 2001). For example with 75 initial genes as described above (fig. 4), the union of all predictor genes found in the top 100 predictors gave a total of 24 genes. These were a subset of the 96 Khan et. al. genes. These are shown in Table 1, (excluding three genes that occur only once among all the predictors). However with the data of fig. 1, we find that out of a total of 25 different genes that comprise all the possible genes used by the 50 predictors, four are different than those found by Khan et. al. Of those four, one of them appears only one time, and two of them occur quite frequently in the predictors. One of these additional genes, neurofibromin 2 appears in all predictors, and the other thioredoxin appears in 37 of the 50 predictors. The third, homeobox B7 appears 6 times. Neurofibromin has been associated with tumorigenesis(Reed & Gutmann, 2001). It is believed that thioredoxin may play a role in cancer and Thioredoxin-1 is often associated with aggressive tumor growth(G.Powis & Montfort, 2001). In a study on multiple carcinogenesis of mouse skin(Chang et al., 1998), Homeobox B7 appears to be expressed at a much lower level than in normal mouse skin. Because this gene only appears in 16% of predictors, this may not be a significant correlation.
Leukemia Data

Microarray data (Golub et al., 1999) was obtained from patients having two types of leukemia, acute lymphoblastic leukemia (ALL), and acute myeloid leukemia (AML). The data here was taken from bone marrow samples and the samples were of different cell types, for example B or T cells and different patient genders. Each sample was analyzed using an Affymetrix microarrays containing expression levels of 7129 genes. The data was divided into 38 training data points and 34 test points.

Using the statistical replication algorithm without deletions, we evolved the predictors and measured the averaged number of misclassifications made as a function of the number of generations. This is done with an initial pool of 50 genes and the results are shown in fig. 6 (lines with crosses). The number of mistakes made by the ensemble of predictors plateaus at about 2. The predictors vary in accuracy; there are predictors that make no mistakes and some that make several. There appears to be no way of distinguishing between them short of using the test data. Data with 200 genes, fig. 6 (solid triangles), shows a similar pattern but does not completely plateau fluctuating in the average number of mistakes from about 1 to three.

Using the deterministic evolution algorithm, we find a much faster convergence to a steady state ensemble of predictors. Using an initial gene pool of 50 and $n_{\text{top}}$ of 100, the number of mistakes goes to about 2 with only three genes in a predictor. This is shown by the open diamonds in fig. 6. The lack of convergence to near perfect predictors is in agreement with other work on this data set (Furey et al., 2000; Li & Yang, 2001; Golub et al., 1999).

Statistical replication with the extra two mutational moves as used in the last section on SRBCT data was performed. Fig. 7 shows the results from starting from the same initial pool of 50 from above. The average number of dimensions in a predictor rises to more than 14 by iteration 63 and then declines, by iteration 80, to a dimension of only 2. During this evolution, the average number of mistakes made on the test data remains fairly constant at 1. Unlike the SRBCT data, there is no convergence to almost perfect prediction, and the individual predictors have a wide range of different dimensions all giving similar predictive ability. Note that although this is the case for the test data, the method predicts perfectly the training data, through leave-one-out cross validation. For example for a pool of 50 genes after iteration 20 the test data prediction is perfect, with an average dimension of about 9.

Varying parameters such as the initial number of genes, $n_{\text{top}}$, and the method of scoring does not lead to a statistically significant improvement in the average number of mistakes made. Also, as the above cases illustrate, the optimum number of genes in a predictors varies between 3 to 25 depending on parameters. This is consistent with recent work on this data where also no clear cutoff in the number of genes needed for an optimal predictor was also found (Li & Yang, 2001).
**Diffuse large B-cell lymphoma**

Recently microarrays in conjunction with supervised learning algorithms were used to study the important problem of Diffuse large B-cell lymphoma (DLBCL), the most common lymphoid malignancy in adults (Shipp et al., 2002). Using 6817 genes from tumor specimens, the authors studied two problems. First, they studied whether their microarray data could be used to distinguish DLBCL from a related B-cell lymphoma, follicular lymphoma (FL). Then they studied if the success or failure of chemotherapy could be predicted from gene expression data of patients. One of the major differences of Shipp et. al. to previous work on the same lymphoma (Alizadeh et al., 2000) is that the earlier work employed unsupervised learning and clustering to predict the putative cell of origin, while the more recent work used supervised learning methods to build a classifier. Here the latter strategy is also followed.

**DLBCL versus FL**

Biopsies from patients before treatment were obtained from 58 patients diagnosed with DLBCL and 19 with FL. Shipp et. al. used leave-one-out cross validation to select their prediction algorithm. They found that a 30 gene predictor could correctly classify 71 of 77 tumors (91%) which corresponds to $P < 10^{-9}$ compared with random prediction.

GESSES was used to analyze the same data using statistical replication with the extra two mutational moves. With different random numbers and different numbers of starting top genes, 77 and 130, GESSES always predicted of 77 out of 77 (100%) of the data correctly. The final predictors ranged in number of genes, from 4 to 12. The 4 gene predictor shared 3 genes in common with those found previously (Shipp et al., 2002). Two of the genes, LDHA Lactate dehydrogenase A and ENO1 Enolase 1 (alpha), are highest ranked by Shipp et. al. as DLBCL “markers”. The gene “(clone GPCR W) G protein-linked receptor gene (GPCR) gene, 5’ end of cds” is ranked 6th as an FL “marker”, and the remaining gene, “protein-coupled receptor (STRL22) mRNA” is not within the top 50 of either class.

It should be noted that leave-one-out cross validation is expected to do better than it would on independent test data (Xing et al., 2001). However the original work (Shipp et al., 2002) did not provide any extra test data, but with 77 subjects it appeared plausible that an independent test could be carried out by splitting the data into two groups, one for test and one for training, to get a more conservative estimate of the predictive value. This was difficult as the data contained only 19 FL samples. Consequently taking out too many of these samples is expected to lead to a substantial degradation in performance of the predictors. Therefore the data was split into 65 training and a 12 test samples. Half of the test data was DLBCL and the other FL. The predictor converged to one with 2 wrong and 10 correctly classified. The number of final dimensions in the predictor was 6. This gives a significance of $P < 1.2 \times 10^{-3}$ compared with random prediction. These numbers are expected to improve with larger data sets.
DLBCL outcome analysis

Shipp *et. al.* went on further to analyze the outcome of chemotherapy. The outcome of 58 patients was divided into two sets, 32 who were “cured” and 26 who were “fatal/refractory”. Using a similar analysis to the DLBCL versus FL work, they used leave-one-out cross validation to select their best predictor. The best predictor they found had 13 genes, and on leave-one-out cross validation they found that it could predict 44 out of 58 (76%) correctly. Again, they did not run their results on an independent test data set.

The same data was analyzed using GESSES utilizing the same parameters as were used above. Starting with the top 130 genes, it was able to find an ensemble of predictors where all outcomes were correctly classified by leave-one-out cross validation. The number of genes for these 86 predictors ranged from a minimum of 22 to a maximum of 31. Out of these predictors, there were a total of 53 separate genes. Three of these genes are identical to ones in the 13 gene predictor of Shipp *et. al.* and one gene, Affymetrix identifier M99436.at, Transducin-like Enhancer Protein 2 may be similar to one used by Shipp *et. al.*, M99435.at, Transducin-like Enhancer Protein 1.

Multiclass diagnosis of Common Tumors

Recent work (Ramaswamy *et al.*, 2001) used microarray data to attempt to distinguish 14 different kinds of tumors. They collected 214 tumor samples spanning these types and analyzed them with an array of different learning algorithms using the expression levels of 16,063 genes. Six different variants of SVM and k nearest neighbor algorithm were tried with different numbers of genes. Leave-one-out cross validation on the 144 subjects used for training data had their best predictor being able to distinguish the correct class on 78% of the samples. When this was tried on the 54 test samples, they also found that it worked on 78% of the test samples. This predictor was an SVM using all 16,063 genes. A slight complication with the analysis is that 8 metastatic samples of different kinds were included in the test data. They found that 6 out of 8 of these were identified correctly. The authors suggest that this indicates that many cancers retain their tissue of origin identity throughout their metastatic development.

The method GESSES uses for filtering the initial top genes, needs to be slightly modified from the method above. This is because there are fairly few samples per class, mostly 8, and $N = 16,063$ initial genes. Therefore by chance, one would expect to have a large number of genes perfectly separating two separate classes even if the data was random. On larger data sets, with more than $2 \log_2(N)$, samples in a class, this would not be a problem. However, the initial filtering algorithm is trivially modified to take this case into account by adding an additional layer of filtering. Instead of distinguishing two types $t_i, t_j$ from each other, one considers the distinguishing $t_i$ from all other types. The algorithm for choosing the initial pool proceeds as before, only now one considers all combinations of $t_i$ and its complement.

A variety of separate runs were done on this data using GESSES with statistical replication and the two extra mutational moves. Excluding metastatic samples, the results typically range from 63% correct.
to 83%. With, metastatic genes included, the results range from 57% (worst with an initial pool of 182) to 80% (best with an initial pool of 273). For example, at the end of a run ($\beta = 681$), while there are many separate predictors (154), 5 predictors classified 12 out of 46 samples incorrectly, 16 predictors made 11 errors, 132 made 10 errors, and 1 predictor made 9 errors.

The number of genes used in these predictors ranged from about 40 to 70. The results from leave-one-out cross validation are considerably higher, typically 92%. As mentioned above, this lower error rate is expected with any method where cross validation is used to select or optimize parameters in a model (Xing et al., 2001). The degree of this bias clearly depends on the details of the algorithm employed.

**Validation**

It is important with any predictive method to understand how success on a set of test data correlates with its true predictive value. An interesting question, which is a major problem in statistics, is to what extent the superiority of one method over another based on a limited data set will hold up with new data.

One way of approaching this problem is to do exchanges of test with training data and then quantify the degree of success a predictor has in these circumstances. This can easily be turned into a quantitative measure for analyzing predictive capability, such as leave-one-out cross validation, but this is still by no means foolproof in the sense that even this measure will still not correlate perfectly with the predictor’s performance on new data.

However one should question the paradigm often employed, which is that one should search for a unique optimal predictor. Given the inherent noise and small number of samples in microarray data, it is not possible to find “the” unique optimal prediction algorithm. As can be seen from this work, there are an enormous number of separate predictors that predict the data perfectly. This is even true of the Leukemia data (Golub et al., 1999) discussed above, where there were still a large number of predictors that predicted all the data perfectly even though on average, the number of mistakes made was about 2.

What this demonstrates is that for the leukemia data, there are large gaps in the characterization of a system’s predictability in algorithms that only yield a single predictor. If we think metaphorically that each member of the ensemble is a predictor chosen by a different researcher, then if a researcher was, by chance, to pick a set of genes that gave perfect predictions on the test data, this could not be said to be superior to another researcher’s predictor who happened by chance to have picked one that made several mistakes. Both predictors are members of an ensemble and have the same statistical weight.

One of the motivations for casting the problem in terms of an ensemble picture is that it gives a much better way of analyzing the above question than methods that attempt to produce a single optimized predictor. Given a set of training data and any other additional information that one wishes to impose, there is a distribution as a function of predictor (i.e. gene set) that gives the probability that this predictor predicts all data perfectly. Although this is difficult to obtain, and many assumptions about
the form of data noise need to be made, the algorithms here attempt to give an approximate sampling of this distribution, in much the same way that Monte Carlo samples the Gibbs distribution.

In summary, given noise in data, there is no unique subset of genes that form an optimal predictor. However there is a unique statistical characterization of subsets. This is equivalent to the traveling salesman problem when the position of the cities has some well characterized uncertainty. There is no unique shortest path, but there is a unique probability distribution of shortest paths.

Now we concentrate on the SRBCT data (Khan et al., 2001), because it can be predicted perfectly and therefore provides a clear means for determining how sensitive the method is to alterations in parameters. We are interested in determining how robust the results are under various alterations.

**Steady State Distribution**

First we investigate the steady state behavior of the ensemble. In order to do this, we allow for all of the kinds of mutations as described in the statistical replication section. We cool the system down as described in the second method on annealing. However we do not let the temperature decrease if the number of distinct systems falls below half of the total number. This stops the system from continuing to cool until it reaches a single distinct state. Instead, the temperature plateaus to a small finite value and the ensemble fluctuates in steady state randomly sampling the distribution of predictors. Fig. 8 shows three quantities as in earlier figures, the average number of mistakes, the number of distinct systems, and the average number of dimensions in a predictor. The system evolves as the temperature decreases until it reaches a constant inverse temperature of $\beta = 2985$ at iteration 168. After this the temperature does not change because the number of distinct systems always remains less than half the total number of systems (75). The number of distinct systems fluctuate as shown (dashed line) and the average number of dimensions of predictors in the ensemble fluctuates around 12. Note that the number of mistakes is almost always zero in this steady state regime. In addition, an examination of the ensembles at different times shows that there is considerable overlap between them.

**Random numbers**

However it is still possible that the ensemble represents some sort of metastable state and that the system is not in equilibrium. To examine the degree to which this ensemble is metastable it is convenient to compare this system with others that have been separately prepared using different random numbers. We prepared six such systems as was done in fig. 5 where the temperature was allowed to decrease almost to zero resulting in an ensemble with at most several distinct predictors. If the ensemble above had been cooled too fast then one would expect little or no overlap with these zero-temperature systems because they were seeded with different random numbers. On the other hand, if many of the predictors cooled to zero-temperature do overlap with the ensemble of predictors, then our ensemble found should be close to the correct equilibrium distribution of predictors.
Fig. 9 shows the six separate runs. The most important feature of all of these is that they end up producing perfect predictors. All curves have similar shapes but differ due to the different random numbers used. The predictors found at the end of all six runs can be found in the steady state ensemble above, with five of the runs having predictors that appear in ensembles in the last iteration and one of them appearing at iteration 189.

This suggests that the ensemble of predictors found is close to equilibrium.

**Scoring parameter**

There is a small constant $C \ll 1$ that is used in the section of the scoring function in part 4, which picks out predictors that best separate out the different types. Here we examine how much these results are affected by changing this constant. Throughout this work (on SRBCT data) we have chosen $C = 0.001$. Here we compare this with $C = 0.0001$. Fig. 10 shows runs with these two constants where we used the second cooling schedule to run the system to almost zero temperature. In both cases the predictors produced predict the test data perfectly and the single predictor found at the end of the $C = 0.0001$ run was one of the three predictors found at the end of the $C = 0.001$ run. In other words the predictors overlapped. The figure shows the average number of dimension as a function of iteration and look similar at the beginning of the run where the constant is irrelevant and at the end of the run where the two systems have converged to similar steady states.

**Discussion**

This paper has described a new and highly effective method, GESSES, that reduces the number of genes necessary to perform an accurate classification. We implemented and tested it, producing an ensemble of predictors that use a minimal number of genes to perform a diagnosis of a cancer from microarray data.

By starting off with an initial pool of candidate genes, an ensemble of predictors is evolved on training data. Each predictor uses a different set of genes and its fitness is scored by analyzing how well it separates the training data into separate classes. The system evolves converging to a set of predictors that can be evaluated using test data.

From the fact that a large number of different gene combinations perform similarly, and the data is still quite noisy, one cannot expect to find the unique combination of genes that is optimum for cancer diagnosis. Clearly there are many combinations that perform very well as they all predict the test data perfectly. However from a practical point of view, the lack of a unique solution does not present a problem. Any one of the the predictors found would be a good starting point for the development of a clinical test based, for example, on antibody assays (He & Friend, 2001).

In addition to the lack of uniqueness, there could very well be other relevant genes that are not employed in the final predictors simply because the initial gene pool is too small. We have found that
even with an initial pool of only 50 genes, it produces many perfect predictors for SRBCT data, utilizing on average less than 15 genes. The initial gene pool was obtained in a way that would likely miss groups of genes that are only predictive when considered together. Of course determining such an initial set of genes is tantamount to solving the problem that was posed here in the first place.

So the question is, if we increase the size of the initial gene pool, do we end up with predictors that use less genes? We have tried an initial pool of 180 genes of which 131 were non-overlapping and it does not lead to a statistically significant decrease in the number of genes, (an average of 12). This is not surprising since even if some of the lower ranked genes form combinations that are, in principle, more parsimonious, they are noisier than the higher ranked genes and hence more genes would be necessary to give a robust predictor. In addition there is often a high correlation of gene expression. Two genes that tend to correlate highly with each other can likely be substituted for each other in a predictor. If this is the case, adding more relevant genes to the initial pool will likely correlate with the ones originally there and therefore their inclusion will not have a large effect on the results.

In addition there are theoretical reasons for limiting the initial pool of genes with wrapper methods(Xing et al., 2001), because with a large initial pools, there is an increased chance of finding good performance on training data by chance alone.

There are many different kinds of prediction algorithms that can be used besides the nearest neighbor algorithm that we chose, among them are artificial neural networks, logistic regression, support vector machines which appear to perform similarly.

We have used a nearest neighbor search method for a variety of reasons. It will classify training points perfectly. It makes little in the way of assumptions of how new data extrapolates from old data. And in conjunction with the replication algorithms used here it is quite efficient because it “learns” rapidly. However which kind of predictor that is used is not the most important part of this work and the replication algorithm could in principle be implemented with any one of the prediction methods mentioned above. Unless prediction methods classify identically, one would expect different prediction methods to give different results, and the poorer the predicting algorithm, the larger the number of genes needed to make a good predictor. It is therefore encouraging to see the k-nearest neighbor algorithm chosen here needing such a small number of genes to predict test data perfectly.

In the case of SRBCT data(Khan et al., 2001), this method was able to find predictors using fewer than 15 genes that were able to reliably classify test data into one of four groups. Some of the genes found were different than the 96 found earlier(Khan et al., 2001) to do this classification and may be of biological significant. The optimum number of genes to use in a predictor is approximately 12 ± 2.

In the case of leukemia data(Golub et al., 1999), less useful information can be obtained. It is probably not possible to use the training data to reliably construct a perfect predictor. It is clear that more data is needed before the same level of prediction can be achieved as with the SRBCT data. This is in accord with other groups findings(Golub et al., 1999; Furey et al., 2000; Li & Yang, 2001). At this point it is not possible to come up with the optimal number of genes needed to predict this data(Li & Yang, 2001).
The main conclusion that one draws from this is that there are many relevant genes in the diagnosis of cancers. However if the data is not complete or is too noisy, it is not possible to exploit this information to its full capacity.

GESSES was also applied successfully to several additional data sets. Two of these are data on Diffuse large B-cell lymphoma (Shipp et al., 2002) and one on the classification of 14 different classes of tumors (Ramaswamy et al., 2001). In these cases, GESSES was able to reduce the number of genes necessary to make a prediction at a given error rate.

For the multiclass tumor data, both GESSES and the initial work of Ramaswamy et. al. fail to achieve 100% success even with the less conservative measure of leave one-out cross validation. The original work gives several reasons for why their experiment is particularly challenging. Among them are the possibility of mis-labeling, the noise in the data, and the small number of examples for each class coupled with the intrinsic biological variation from specimen to specimen.

It is hoped that using GESSES could help lead to practical uses of microarray data in cancer diagnosis, for example using antibody assays (He & Friend, 2001) from the handful of genes found in this work.

Acknowledgments

The author thanks Francoise Chamut, James Conklin, and David Draper for useful discussions.
References


Figure 1: The average number of genes, $\langle \text{dim} \rangle$, for an ensemble of predictors as a function of the number of iterations (or generations), for SRBCT data (Khan et al., 2001) is shown with the + points. The average number of mistakes made is shown by the dashed line with the $\times$ points. The number of genes used here was 50, and the algorithm used was a statistical replication algorithm with only additions ($n_m = 2$).
Figure 2: The statistical algorithm using all mutational moves. The average number of dimensions (solid line with + symbols), the average number that the predictor got wrong (× symbols), and the number of distinct systems in the ensemble (dashed lines) as a function of the number of iterations. The number of genes used here was 50 (n_m = 20).
Figure 3: The average number of genes, $<\text{dim}>$, for an ensemble of predictors as a function of the number of iterations (or generations), for SRBCT data (Khan et al., 2001) is shown with the + points. The average number of mistakes made is shown by the dashed line with the $\times$ points. The number of genes used here was 50, and the algorithm used was deterministic described in the text.
Figure 4: The average number of genes and average number of mistakes made as a function of iteration in a predictor generated by the deterministic algorithm for an initial pool of 75 genes SRBCT data (Khan et al., 2001). The parameters are described in the text.
Figure 5: The statistical algorithm using all mutational moves, which includes deletions with an initial pool of 75 genes for the SRBCT data (Khan et al., 2001). The average number of dimensions (solid line with + symbols), the average number that the predictor got wrong (× symbols), and the number of distinct systems in the ensemble (dashed lines) as a function of the number of iterations ($n_m = 20$).
Figure 6: The average number of mistakes for an ensemble of predictors as a function of the average number of genes, for the leukemia data (Golub et al., 1999). Results for two algorithms are shown here. First, the statistical replication algorithm with no deletion ($n_m = 2$). The number of genes used here was 200 (solid triangles) and 50 (lines with crosses). Note the curve is not singled valued because as the predictor evolves, the average number of genes and number of mistakes can increase due to statistical fluctuations. Second, the deterministic algorithm with 50 genes (open diamonds).
Figure 7: The statistical algorithm using all mutational moves, which includes deletions with an initial pool of 50 genes for the leukemia data (Golub et al., 1999). The average number of dimensions (top curve, solid line with + symbols), the average number that the predictor got wrong (bottom curve, × symbols), as a function of the number of iterations.
Figure 8: A steady state run of the statistical algorithm with deletions as described in the text using SRBCT data (Khan et al., 2001). The average number of dimensions is the top curve. The average number of wrong predictions is the bottom curve. The predictors give almost perfect prediction after 100 iterations. The average number of distinct systems is the curve in the middle ($n_m = 20$).
Figure 9: Six runs of the statistical algorithm with deletions as described in the text, using different random numbers using SRBCT data (Khan et al., 2001). The average number of dimensions in a predictor is represented by lines with points, the average number of wrong predictions is represented only by lines. All six predictors asymptote to almost perfect prediction ($n_m = 20$).
Figure 10: Two runs of the statistical algorithm with deletions as described in the text, using different constants $C$ for SRBCT data (Khan et al., 2001). The curve which is predominantly higher represents $C = 0.0001$, and the other code is $C = 0.001$ ($n_m = 20$).
<table>
<thead>
<tr>
<th>id#</th>
<th>gene description</th>
</tr>
</thead>
<tbody>
<tr>
<td>365826</td>
<td>growth arrest-specific 1</td>
</tr>
<tr>
<td>298062</td>
<td>&quot;troponin T2, cardiac&quot;</td>
</tr>
<tr>
<td>383188</td>
<td>recoverin</td>
</tr>
<tr>
<td>296448</td>
<td>insulin-like growth factor 2 (somatomedin A)</td>
</tr>
<tr>
<td>769959</td>
<td>&quot;collagen, type IV, alpha 2&quot;</td>
</tr>
<tr>
<td>377461</td>
<td>&quot;caveolin 1, caveolae protein, 22kD&quot;</td>
</tr>
<tr>
<td>325182</td>
<td>&quot;cadherin 2, N-cadherin (neuronal)&quot;</td>
</tr>
<tr>
<td>1473131</td>
<td>&quot;transducin-like enhancer of split 2, homolog of Drosophila E(sp1)&quot;</td>
</tr>
<tr>
<td>207274</td>
<td>Human DNA for insulin-like growth factor II (IGF-2); exon 7 and additional ORF</td>
</tr>
<tr>
<td>357031</td>
<td>&quot;tumor necrosis factor, alpha-induced protein 6&quot;</td>
</tr>
<tr>
<td>812105</td>
<td>transmembrane protein</td>
</tr>
<tr>
<td>241412</td>
<td>E74-like factor 1 (ets domain transcription factor)</td>
</tr>
<tr>
<td>183337</td>
<td>&quot;major histocompatibility complex, class II, DM alpha&quot;</td>
</tr>
<tr>
<td>796258</td>
<td>&quot;sarcoglycan, alpha (50kD dystrophin-associated glycoprotein)&quot;</td>
</tr>
<tr>
<td>866702</td>
<td>&quot;protein tyrosine phosphatase, non-receptor type 13 (APO-1/CD95 (Fas)-associated phosphatase)&quot;</td>
</tr>
<tr>
<td>770394</td>
<td>&quot;Fc fragment of IgG, receptor, transporter, alpha&quot;</td>
</tr>
<tr>
<td>52076</td>
<td>olfactomedinrelated ER localized protein</td>
</tr>
<tr>
<td>609663</td>
<td>&quot;protein kinase, cAMP-dependent, regulatory, type II, beta&quot;</td>
</tr>
<tr>
<td>814260</td>
<td>follicular lymphoma variant translocation 1</td>
</tr>
<tr>
<td>784224</td>
<td>fibroblast growth factor receptor 4</td>
</tr>
<tr>
<td>295985</td>
<td>ESTs</td>
</tr>
</tbody>
</table>

Table 1: Genes found that perfectly predict SRBCT samples