Tissue Classification with Gene Expression Profiles*†

Amir Ben-Dor‡
Agilent Laboratories
3500 Deer Creek Road, MS 25U-5
Palo Alto, CA 94304
and
The University of Washington, Seattle
amirbd@cs.washington.edu

Laurakay Bruhn
Agilent Laboratories
3500 Deer Creek Road, MS 25U-7
Palo Alto, CA 94304
laurakay_bruhn@agilent.com

Nir Friedman
School of Computer Science & Engineering
Hebrew University
Jerusalem, 91904, ISRAEL
nir@cs.huji.ac.il

Iftach Nachman
Center for Computational Neuroscience
and
School of Computer Science & Engineering
Hebrew University
Jerusalem, 91904, Israel
iftach@cs.huji.ac.il

Michèl Schummer
University of Washington
Box 357730
Seattle, WA 98195
michel@drschummer.de

Zohar Yakhini
Agilent Laboratories
MATAM Bdg 30
Haifa 31905, Israel
zohar_yakhini@agilent.com

May 3, 2000

Abstract

Constantly improving gene expression profiling technologies are expected to provide understanding and insight into cancer related cellular processes. Gene expression data is also

*A preliminary version of this work appeared in *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology*, 2000.

†Part this work was done while Amir Ben-Dor was at University of Washington, Seattle, with support from the *Program for Mathematics and Molecular Biology (PMMB)*. Nir Friedman and Iftach Nachman were supported through the generosity of the Michael Sacher Trust and Israeli Science Foundation equipment grant. Part of this work was done while Zohar Yakhini was visiting the Computer Science Department at the Technion.

‡Contact author.

expected to significantly aid in the development of efficient cancer diagnosis and classification platforms. In this work we examine three sets of gene expression data measured across sets of tumor(s) and normal clinical samples: The first set consists of 2,000 genes, measured in 62 epithelial colon samples (Alon et al. 1999). The second consists of $\approx$ 100,000 clones, measured in 32 ovarian samples (unpublished extension of data set described in (Schummer et al. 1999)). The third set consists of $\approx$ 7,100 genes, measured in 72 bone marrow and peripheral blood samples (Golub et al. 1999). We examine the use of scoring methods, measuring separation of tissue type (e.g., tumors from normals) using individual gene expression levels. These are then coupled with high dimensional classification methods to assess the classification power of complete expression profiles. We present results of performing *leave-one-out cross validation* (LOOCV) experiments on the three data sets, employing *nearest neighbor* classifier, *SVM* (Cortes & Vapnik 1995), *AdaBoost* (Freund & Schapire 1997) and a novel clustering based classification technique. As tumor samples can differ from normal samples in their cell-type composition we also perform LOOCV experiments using appropriately modified sets of genes, attempting to eliminate the resulting bias.

We demonstrate success rate of at least 90% in tumor vs normal classification, using sets of selected genes, with as well as without cellular contamination related members. These results are insensitive to the exact selection mechanism, over a certain range.

1 Introduction

The process by which the approximately 100,000 genes encoded by the human genome are expressed as proteins involves two steps. DNA sequences are initially transcribed into mRNA sequences. These mRNA sequences in turn are translated into the amino acid sequences of the proteins that perform various cellular functions. A crucial aspect of proper cell function is the regulation of gene expression, so that different cell types express different subsets of genes. Measuring mRNA levels can provide a detailed molecular view of the subset of genes expressed in different cell types under different conditions. Recently developed array-based methods enable simultaneous measurements of the expression levels of thousands of genes. These measurements are made by quantitating the hybridization (detected for example, by fluorescence) of cellular mRNA to an array of defined cDNA or oligonucleotide probes immobilized on a solid substrate. Array methodologies have led to a tremendous acceleration in the rate at which gene expression pattern information is accumulated (DeRisi. et al. 1997, Khan et al. 1998, Lockhart et al. 1996, Wen et al. 1998). Measuring gene expression levels under different conditions is important for expanding our understanding of gene function, how various gene products interact, and how experimental treatments can affect cellular function.

Gene expression data can help in better understanding of cancer. Normal cells can evolve into malignant cancer cells through a series of mutations in genes that control the cell cycle, apoptosis, and genome integrity, to name only a few. As determination of cancer type and stage is often crucial to the assignment of appropriate treatment (Golub et al. 1999), a central goal of the analysis of gene expression data is the identification of sets of genes that can serve, via expression profiling assays, as classification or diagnosis platforms.

Another important purpose of gene expression studies is to improve understanding of cellular responses to drug treatment. Expression profiling assays performed before, during and after treatment, are aimed at identifying drug responsive genes, indications of treatment outcomes, and at identifying potential drug targets (Clarke et al. 1999). More generally, complete profiles can be considered as a potential basis for classification of treatment progression or other trends in the evolution of the treated cells.

Data obtained from cancer related gene expression studies typically consists of expression level measurements of thousands of genes. This complexity calls for data analysis methodologies that will efficiently aid in extracting relevant biological information. Previous gene expression analysis work emphasizes clustering techniques, which aim at partitioning the set of genes into subsets that are expressed similarly across different conditions. Indeed, clustering has been demonstrated to identify functionally related families of genes (Ben-Dor et al. 1999, DeRisi. et al. 1997, Chu et al. 1998, Eisen et al. 1998, Iyer et al. 1999, Wen et al. 1998). Similarly, clustering methods can be used to divide a set of cell samples into clusters based on their expression profile. In (Alon et al. 1999) this approach was applied to a set of colon samples which was divided into two groups, one containing mostly tumor samples, and the other containing mostly normal tissue samples.

Clustering methods, however, do not use any tissue annotation (e.g., tumor vs. normal) in the partitioning step. This information is only used to asses the success of the method. Such methods are often referred to as *unsupervised*. In contrast, *supervised* methods, attempt to predict the classification of new tissues, based on their gene expression profiles after training on examples that have been classified by an external "supervisor".

The purpose of this work is to rigorously assess the potential of classification approaches based on gene expression data. We present a novel clustering based classification methodology, and apply it together with two other recently developed classification approaches, *Boosting* (Schapire 1990, Freund & Schapire 1997) and *Support Vector Machines* (Cortes & Vapnik 1995, Vapnik 1999) to three data sets. These sets involve corresponding tissue samples from tumor and normal biopsies. The first is a data set of colon cancer (Alon et al. 1999), the second is a data set of ovarian cancer (an extension of the data set reported in (Schummer et al. 1999)), and the third is a data set of leukemia (Golub et al. 1999). We use established statistical tools, such as *leave one out cross validation* (LOOCV), to evaluate the predictive power of these methods in the data sets.

One of the major challenges of gene expression data is the large number of genes in the data sets. For example, one of our data sets includes almost 100,000 clones. Many of these clones are not relevant to the distinction between cancer and tumor and introduce noise in the classification process. Moreover, for diagnostic purposes it is important to find small sets of genes that are sufficiently informative to distinguish between cells of different types. To this end we suggest a simple combinatorial error rate score for each gene, and use this method to select informative genes. As we show, selecting relatively small subsets of genes can drastically improve the performance. Moreover, this selection process also isolates genes that are potentially intimately related to the tumor makeup and the pathomechanism.

To realistically assess the performance of such methods one needs to address the issue of *sample contamination*. Tumor and normal samples may dramatically differ in terms of their cell-type composition. For example, in the colon cancer data (Alon et al. 1999), the authors observed that the normal colon biopsy also included smooth muscle tissue from the colon walls. As a result, smooth muscle related genes showed high expression levels in the normal samples compared to the tumor samples. This artifact, if consistent, could contribute to success in classification. To eliminate this effect we remove the muscle specific genes and observe the effect on the success rate of the process.

The rest of the paper is organized as follows. In Section 2, we describe the principal classification methods we use in this study. These include two state of the art methods from machine learning, and a novel approach based on the clustering algorithm of (Ben-Dor et al. 1999). In Section 3, we describe the three data sets, the LOOCV evaluation method, and evaluate the classification methods on the three data sets. In Section 4 we address the problem of gene selection. We propose a simple method for selecting informative genes and evaluate the effect of gene selection on the classification methods. In Section 5, we examine the effect of sample contamination on possible classification. We conclude in Section 6 with a discussion of related work and future directions.

2 Classification Methods

In this section, we describe the main classification methods that we will be using in this paper. We start by formally defining the classification problem. Assume that we are given a *training set* D, consisting of pairs $\langle x_i, l_i \rangle$, for $i = 1, \ldots, m$. Each *sample* x_i is a vector in $\mathbf{R}^N$ that describes expression values of N genes/clones. The *label* l_i associated with x_i is either -1 or $+1$ (for simplicity, we will discuss two-label classification problems). A classification algorithm is a function f that depends on two arguments, the training set D, and a query $x \in \mathbf{R}^N$, and returns a predicted label $\hat{l} = f_D(x)$. We also allow for no classification to occur if x is either close to none of the classes or when it is too borderline for a decision to be taken. Formally, this is realized by allowing $\hat{l}$ to be -1 , $+1$ or 0, the latter representing an unclassified query. Good classification procedures predict labels that typically match the "true" label of the query. For a precise definition of this notion in the absence of the unclassified option assume that there is some (unknown) joint distribution $P(x, l)$ of expression patterns and labels. The *error* of a classification function $f_D(\cdot)$ is defined as $P(f_D(x) \neq l)$. Of course, since we do not have access to $P(\cdot)$, we cannot precisely evaluate this term and use estimators instead. When unclassified is accepted as a possible output one needs to consider the costs/penalties of the various outcomes in analyzing the value of a classification method. For a comprehensive discussion of classification problems see (Bishop 1995, Duda & Hart 1973, Ripley 1996).

2.1 Nearest Neighbor Classifier

One of the simplest classification procedures is the *nearest neighbor* classifier (Duda & Hart 1973). The intuition is simple. To classify a query x, find the most similar example in D and predict that x has the same label as that example. To carry out this procedure we need to define a similarity measure on expression patterns. In our experiments, we use the *Pearson correlation* as a measure of similarity (see, e.g., (Eisen et al. 1998)).

Formally, let

$$k_p(x, y) = \frac{E[(x_i - E[x])(y_i - E[y])]}{\sqrt{Var[x]\, Var[y]}}$$

be the Pearson correlation between two vectors of expression levels. Given a new vector x, the nearest neighbor classification procedure searches for the vector x_i in the training data that maximizes $k_p(x, x_i)$, and returns l_i, the label of x_i.

This simple non-parametric classification method does not take any global properties of the training set into consideration. However, it is surprisingly effective in many types of classification problems. We use it in our analysis as a strawman, to which we compare the more sophisticated classification approaches.

2.2 Using Clustering for Classification

Recall that clustering algorithms, when applied to expression patterns, attempt to partition the set of elements into clusters of patterns, so that all the patterns within a cluster are similar to each other, and different from patterns in other clusters. This suggests that if the labeling of patterns is correlated with the patterns, then unsupervised clustering of the data (labels not taken into account) would cluster patterns with the same label together and separate patterns with different labels. Indeed, such a result is reported by Alon et al. (1999) in their analysis of colon cancer. Their study (which we describe in more detail in Section 3) involves gene expression patterns from colon samples that include both tumors and normal tissues. Applying a hierarchical clustering procedure to the data, Alon et al. observe that the topmost division in the dendrogram divides samples into two groups, one predominantly tumor, and the other predominantly normal. This suggests that for some types of classification problems, such as tumor vs. normal, clustering can distinguish between labels. Following this intuition, we build a clustering based classifier. We first describe the underlying clustering algorithm and then present the classifier.

2.2.1 The clustering algorithm

The CAST algorithm, implemented in the BioClust analysis software package (Ben-Dor et al. 1999), takes as input a threshold parameter t, which controls the granularity of the resulting cluster structure, and a similarity measure between the tissues.[1] We say that a tissue v has *high similarity* to a set of tissues $\mathcal{C}$, if the average similarity between v and the tissues in $\mathcal{C}$ is at least t. Otherwise, we say that v has *low similarity* to $\mathcal{C}$. CAST constructs the clusters one at a time, and halts when all tissues are assigned to clusters. Intuitively, the algorithm alternates between adding high similarity tissues to $\mathcal{C}$, and removing low similarity tissues from it. Eventually, all the tissues in $\mathcal{C}$ have high similarity to $\mathcal{C}$, while all the tissues outside of $\mathcal{C}$ have low similarity to $\mathcal{C}$. At this stage the cluster $\mathcal{C}$ is closed, and a new cluster is started (See (Ben-Dor et al. 1999) for complete description of the algorithm).

Clearly, the threshold value t, has great effect on the resulting cluster structure. As t increases, the clusters formed are smaller. At the extreme case, if t is high enough, each tissue would form a singleton cluster. Similarly, as t decreases, the clusters tend to get larger. If t is low enough, all tissues are assigned to the same cluster.

[1] In this work we use the Pearson correlation between gene expression profiles as the similarity measure. However, any similarity measure can be used.

2.2.2 Clustering based classification

As described above, the threshold parameter t determines the cohesiveness and the number of the resulting clusters. A similar situation occurs in other clustering algorithms. For example, in hierarchical clustering algorithms (e.g., (Alon et al. 1999, Eisen et al. 1998)), the cutoff "level" of the tree controls the number of clusters. In any clustering algorithm, it is clear that attempting to partition the data into exactly two clusters will not be the optimal choice for predicting labels. For example, if the tumor class consists of several types of tumors, then the most noticeable division into two clusters might separate "extreme" tumors from the milder ones and the normal tissues, and only a further division will separate the normals from the milder tumors.

For the purpose of determining the right parameter to be used in clustering data that contains some labeled samples we propose a measure of cluster structure *compatibility* with a given label assignment. The intuition is simple: on the one hand, we want clusters to be uniformly labeled and therefore penalize pairs of samples that are within the same cluster but have different labels; on the other hand, we do not want to create unnecessary partitions and therefore penalize pairs of samples that have the same label, but are not within the same cluster.

Formally, we define the *compatibility* score of a cluster structure with the training set as the sum of two terms. The first is the number of tissue pairs (v, u) such that v and u have the same label, and are assigned to the same cluster. The second term is the number of (v, u) pairs that have different labels, and are assigned to different clusters. This score is also called the *matching coefficient* in the literature (Everitt 1993). To handle label assignments defined only on a subset of the data we restrict the comparison to count pairs of examples for which labels are assigned (the matching coefficient for a submatrix is computed).

Using this notion, we can optimize, using a binary search, the choice of clustering parameters to find the most compatible clustering. That is: we consider different threshold values, t; use CAST to cluster the tissues; measure the compatibility $C(t)$ of the resulting cluster structure with the given label assignment; and finally, choose the clustering that has maximal $C(t)$. Thus, although the clustering algorithm is *unsupervised*, in the sense that it does not take into account the labels, we use a supervised procedure for choosing the clustering threshold. We also emphasize that this general idea can be applied to any parameter dependent clustering method, and is not restricted to our particular choice.

To classify a query sample we cluster the training data and the query, maximizing compatibility to the labeling of the training data. We then examine the labels of all elements of the cluster the query belongs to and use a simple majority rule to determine the unknown label. The intuition is that the query's label should agree with the prevailing label in its cluster. Various majority rules, taking into account statistical confidence can be used. When confidence is too low the query is labeled as *unclassified*. The stringency of this test determines the strictness of our classification rule. In the current experiment we use the most liberal rule, i.e. a query is unclassified only if there is an equal number of elements of each label in its cluster. The choice of majority rule depends on the cost of non-classification vs. the cost of misclassification.

2.3 Large-Margin Classifiers

The cluster-based approach we discussed in the previous section attempts to find inherent structure in the data (i.e., clusters of samples) and uses this structure for prediction. We can also use *direct* methods that attempt to learn a *decision surface* that separates the positive labeled samples from the negatively labeled samples.

The literature of supervised learning discusses a large number of methods that learn decision surfaces. These methods can be described by two aspects. First, the class of surfaces from which one is selected. This question is often closely related to the *representation* of the learned surface. Examples include linear separation (which we discuss in more detail below), decision-tree representations, and two-layer artificial neural networks. Second, the learning rule that is being used. For example, one of the simplest learning rules attempts to minimize the number of errors on the training set.

Application of direct methods to our domain can suffer from a serious problem. In gene-expression data we expect N, the number of measured genes, to be significantly larger than M, the number of samples. Thus, due to the large number of dimensions there are many simple decision surfaces that can separate the positive examples from the negative ones. This means that counting the number of training set errors is not restrictive enough to distinguish good decision surfaces from bad ones (in terms of their performance on examples not in the training set).

In this paper, we use two methods that received much recent attention in the machine learning literature. Both methods attempt to follow the intuition that classification of examples depends not only on the region they are in, but also on a notion of *margin*: how close they are to the decision surface. Classification of examples with small margins is not as confident as classification of examples with large margins. (Given slightly different training data, the estimated decision surface moves a bit, thus changing the classification of points which are close to it.) This reasoning suggests that we should select a decision surface that classifies all the training examples correctly with large margin. Following the same argument, given the learned decision surface and an unlabeled sample x, we can set a threshold on the margin of x for classification. If x is closer to the surface then the allowed threshold, we mark is as unclassified. Again, the threshold will depend on the relative costs of the different outcomes.

The basic intuition of large margin classifiers is developed in quite different manners in the following two approaches.

2.3.1 Support Vector Machines

Support vector machines (SVM) were developed in (Cortes & Vapnik 1995, Vapnik 1999). A tutorial on SVMs can be found in (Burges 1998). The intuition for support vector machines is best understood in the example of linear decision rules. A linear decision rule can be represented by a hyperplane in R^N such that all examples on the one side of the hyperplane are labeled positive and all the examples on the other side are labeled negative. Of course, in sufficiently high-dimensional data we can find many linear decision rules that separate the examples. Thus, we want to find a hyperplane that is as far away as possible from all the examples. More precisely, we want to find a hyperplane that separates the positive examples from the negative ones, and also maximizes the minimum distance of the closest points to the hyperplane.

We now make this intuition more concrete. A linear decision rule can be represented by a hyperplane in R^N such that all examples on the one side of the hyperplane are labeled positive and all the examples on the other side are labeled as negative. Such a rule can be represented by a vector $w \in R^N$ and a scalar b that together specify the hyperplane $w \cdot x + b = 0$. Classification for a new example x is performed by computing $\text{sign}(w \cdot x + b)$. Recall that $|\frac{x \cdot w + b}{||w||}|$ is the distance from x to the line $x \cdot w + b = 0$. Thus, if all points in the training data satisfy

$$l_i(x_i \cdot w + b) \geq 1 \tag{1}$$

than all points are correctly classified, and all of them have a distance of at least $1/||w||$ from the hyperplane. We can find the hyperplane that maximizes the margin of error by solving the following quadratic program:

Minimize $||w||^2$
Subject to $l_i(x_i \cdot w + b) \geq 1$ for $i = 1, \ldots, m$.

Such quadratic programs can be solved in the dual form. This dual form is posed in terms of auxiliary variables α_i. The solution has the property that

$$w = \sum_i \alpha_i l_i x_i,$$

and thus, we can classify a new example x by evaluating

$$\text{sign}(\sum_i \alpha_i l_i \langle x_i, x \rangle + b) \tag{2}$$

In practice, there is a range of optimization methods that can be used for solving the dual optimization problem. See Burges (1998) for more details.

The SVM dual optimization problem and its solution have several attractive properties. Most importantly, only a subset of the training examples determine the position of the hyperplane. Intuitively, these are exactly these samples that are at a distance $1/||w||$ from the hyperplane. It turns out that the dual problem solution assigns $\alpha_i = 0$ to all examples that are not "supporting" the hyperplane. Thus, we only need to store the *support vectors* x_i for which $\alpha_i > 0$. (Hence the name of the technique.)

It is clear that linear hyperplanes are a restricted form of decision surfaces. One method of learning more expressive separating surfaces is to project the training examples (and later on queries) into a higher-dimensional space, and learn a linear separator in that space. For example, if our training examples are in R^1, we can project each input value x to the vector (x, x^2). A linear separator in the projected space is equivalent to an interval in the original representation of the training examples.

In general, we can fix an arbitrary projection $\Phi : \mathbf{R}^N \mapsto \mathbf{R}^M$ to higher dimensional space, and get more expressive decision surfaces. However, it seems that it requires us to solve a harder SVM optimization problem, since we now deal with higher dimensional space.

Somewhat surprisingly, it turns out that the dual form of the quadratic optimization problem involves only inner products of vectors in R^N (i.e., expressions of the form $\langle \Phi(x_i), \Phi(x_j) \rangle$). In other words, vectors x_i do not appear outside the scope of an inner product operation. Similarly, the classification rule (2) only examines vectors in R^N inside the inner product operation. Thus, if we want to consider any projection $\Phi : \mathbf{R}^N \mapsto \mathbf{R}^M$, we can find an optimal separating hyperplane in that projected space by solving the quadratic problem with inner products $\langle \Phi(x_i), \Phi(x_j) \rangle$. Thus, if we can compute these inner products, the cost of the quadratic optimization problem does not increase with the number of dimensions in the projected space.

Moreover, for many projections there are *kernel* functions that compute the result of the inner product. A *kernel function* k for a projection Φ satisfies $k(x, y) = \langle \Phi(x), \Phi(y) \rangle$. Given a legal kernel function, we can use it without knowing or explicitly computing the actual mapping Φ. For many projections, the kernel function can be computed in time that is linear in N, regardless of the dimension M. For example, kernel functions of the form $k(x, y) = (\langle x, y \rangle + 1)^p$ compute dot-products for a projection from R^N to $R^{\binom{N}{p}}$, where each coordinate is a polynomial of degree p in the original coordinates in R^N. Note that in this example, a linear decision surface in the projected space corresponds to a polynomial manifold of degree p in the original space.

To summarize, if we want to learn expressive decision surfaces, we can choose a kernel function, and use it instead of the inner-product in the execution of the SVM optimization. This is equivalent to learning a linear hyperplane in the projected space.

In this work we consider two kernel functions:

- The linear kernel $k_1(x, y) = \langle x, y \rangle$.
- The quadratic kernel $k_2(x, y) = (\langle x, y \rangle + 1)^2$.

The rational for using these simple kernels, is that since our input space is high dimensional, we can hope to find a simple separation rule in that space. We therefore test the linear separator, and the next order separator as a comparison to check if higher order kernels can yield better results.

Note that the quadratic kernel is strictly more expressive than the linear one: any decision surface that can be represented with $k_1(\cdot, \cdot)$ can also be represented with $k_2(\cdot, \cdot)$. Nonetheless, it is not obvious that the more expressive representation will always perform better. Given a larger set of decision surfaces to choose from, this procedure is more susceptible to *overfitting*, i.e. learning a decision surface that performs well on the training data but performs badly on test data.

2.3.2 Boosting

Boosting was initially developed as a method for constructing good classifiers by repeated calls to "weak" learning procedures (Freund & Schapire 1997, Schapire 1990). The assumption is that we have access to a "weak learner" that given a training set D, constructs a function $f_D(x)$. The learner is weak in the sense that the *training set* error is only slightly better than that of a random guess. Formally, we assume that $f_D(x)$ classifies at least $1/2 + 1/poly(m)$ of the input space correctly.

In this paper, we use a fairly simple weak learner, that finds simple rules of the form:

$$g(x : j, t, d) = \begin{cases} d & x[j] > t \\ -d & x[j] < t \end{cases}$$

where x is an expression profile (e.g., a tissue to be classified), j is an index of a gene, $x[j]$ is the expression value of the j'th gene in the vector x, t is a threshold corresponding to gene j, and $d \in \{+1, -1\}$ is a direction parameter. Such a rule is called a *decision stump*. Given a data set D, we learn decision stumps by exhaustively searching all genes, for each gene searching over all thresholds and directions, and finally returning the combination that has the smallest number of errors.[2]

Boosting uses the weak learning procedure (in our case, the decision stump learner) to construct a sequence of classifiers $f_1, \ldots, f_k$, and then uses a weighted vote among these classifiers. Thus, the prediction made by the boosting algorithm has the form :

$$h(x) = \text{sign}(\sum_j w_j f_j(x))$$

where w_i are the weights assigned to the classifiers.

The crux of the algorithm is the construction of the sequence of classifiers. The intuition is simple. Suppose that we train the weak learner on the original training data D to get a classifier $f_1(x)$. Then, we can find the examples in D that are classified incorrectly by f_1. We want to force the learning algorithm to give these examples special attention. This is done by constructing a new training data set in which these examples are given more weight. Boosting then invokes the weak learner on the reweighted training set and obtains a new classifier. Examples are then reweighted again, and the process is iterated. Thus, boosting adaptively reweights training examples to focus on the "hard" ones.[3] In this paper, we use the AdaBoost algorithm of Freund and Schapire (1997). This algorithm is described in Figure 1.

In practice boosting is an efficient learning procedure that usually has a small number of errors on test sets. The theoretical understanding of this phenomenon uses a notion of margin that is quite similar to the one defined for SVMs. Recall, that boosting classification is made by averaging the "votes" of many classifiers. Define the margin of example x_i to be

$$m_i = l_i \sum_j w_j f_j(x_i).$$

By definition, we have that if $m_i > 0$, then $h(x_i) = l_i$, and thus x_i is classified correctly. However, if m_i is close to 0, then this classification is "barely" made. On the other hand, if m_i is close to 1, then a large majority of the classifiers make the right prediction on x_i. The analysis of Schapire et al. (1998) and Mason et al. (1999) shows that the generalization error of boosting (and other voting schemes) depends on the distribution of margins of training examples. Schapire et al. also show that repeated iterations of AdaBoost continually increase the smallest margin of training examples. This is contrasted with other voting schemes that are not necessarily increasing the margin for the training set examples.

3 Evaluation

In the previous section we discussed several approaches for classification. In this section we examine their performance, on experimental data.

[2]Note that for each gene, we need to consider only m rules, since the gene takes at most m different values in the training data. Thus, we can limit our attention to mid-way points between consecutive values attained by the j'th gene in the training data.

[3]More precisely, boosting distorts the distribution of the input samples. For some weak learners, like the stump classifier, this can be simulated by simply reweighting the samples.

Input:

- A data set of m labeled examples $\{(x_1, l_1), \ldots, (x_m, l_m)\}$
- A weak learning algorithm L.

Initialize: the distribution over the data set:

$$D_1(x_i) = 1/m \text{ for } i = 1 \ldots m$$

Iterate: for $t = 1, 2, ..., T$

- Call L with distribution D_t; Get back a hypothesis h_t.
- Calculate the error of h_t:

$$\epsilon_t = \sum_{i=1}^{m} D_t(x_i) 1\{l_i \neq h_t(x_i)\}$$

- Set $w_t = \frac{1}{2} log \frac{1-\epsilon_t}{\epsilon_t}$
- Set the new distribution to be:

$$D_{t+1}(x_i) \propto D_t(x_i) e^{w_t l_i h_t(x_i)}$$

  such that D_{t+1} will sum to 1.

Output: The final hypothesis

$$h(x) = sign(\sum_{t=1}^{T} w_t h_t(x))$$

Figure 1: The generic AdaBoost algorithm. In our setting we use the weak learner L is a procedure that searches for a decision stump that has the smallest (weighted) error on the training data (with weights defined by D_t). The final output of AdaBoost in this case is a weighted combination of decision stumps.

3.1 Data Sets

Descriptions of the three data sets studied follow. The first two data sets involve comparing tumor and normal samples of the same tissue, while the third data set involves samples from two variants of the same disease.

Colon cancer data set. This data set is a collection of expression measurements from colon biopsy samples reported by Alon et al. (1999). The data set consists of 62 samples of colon epithelial cells. These samples were collected from colon-cancer patients. The "tumor" biopsies were collected from tumors, and the "normal" biopsies were collected from healthy parts of the colons of the same patients. The final assignments of the status of biopsy samples were made by pathological examination.

Gene expression levels in these 62 samples were measured using high density oligonucleotide arrays. Of the ≈ 6000 genes represented in these arrays, 2000 genes were selected based on the confidence in the measured expression levels. The data, 62 samples over 2000 genes is available at `http://www.molbio.princeton.edu/colondata`.

Ovarian cancer data set. This data set is a collection of expression measurements from 32 samples: 15 ovary biopsies of ovarian carcinomas, 13 biopsies of normal ovaries, and 4 samples of other tissues. Thus, the data set consists of 28 samples labeled as tumor or normal. Gene expression levels in these 32 samples were measured using a membrane-based array with radioactive probes. The array consisted of cDNAs representing approximately 100,000 clones from ovarian clone libraries. For some of the samples, there are two or three repeated hybridizations for error assessments. In these cases, we collapsed the repeated experiments into one experiment, represented by the average level of expression.

Leukemia data set. This data set is a collection of expression measurements reported by Golub et al. (1999). The data set contains 72 samples. These samples are divided to two variants of leukemia: 25 samples of *acute myeloid leukemia* (AML) and 47 samples of *acute lymphoblastic leukemia* (ALL). The source of the gene expression measurements was taken from 63 bone marrow samples and 9 peripheral blood samples. Gene expression levels in these 72 samples were measured using high density oligonucleotide microarrays. The expression levels of 7129 genes are reported. The data, 72 samples over 7129 genes is available at `http://www.genome.wi.mit.edu/MPR`.

3.2 Estimating Prediction Errors

When evaluating the prediction accuracy of the classification methods we described above, it is important not to use the *training error*. Most classification methods will perform well on examples they have seen during training. To get a realistic estimate of performance of the classifier, we must test it on examples that did not appear in the training set. Unfortunately, since we have a small number of examples, we cannot remove a sizeable portion of the training set and use it for testing.

A common method to test accuracy in such situations is *cross-validation*. To apply this method, we partition the data into k sets of samples, $C_1, \ldots, C_k$ (typically, these will be of roughly the same size). Then, we construct a data set $D_i = D - C_i$, and test the accuracy of $f_{D_i}()$ on the samples in C_i. Having done this for all $1 \leq i \leq k$ we estimate the accuracy of the method by averaging the accuracy over the k cross-validation trials.

Cross-validation has several important properties . First, the training set and the test set in each trial are disjoint. Second, the classifier is tested on each sample exactly once. Finally, the training set for each trial is $(k-1)/k$ of the original data set. Thus, for large k, we get a relatively unbiased estimate of the classifier behavior given a training set of size m.

There are several possible choices of k. A common approach is to set $k = m$. In this case, every trial removes a single sample and trains on the rest. This method is known as *leave one out cross validation* (LOOCV). Other common choices are $k = 10$ or $k = 5$. LOOCV has been in use since early days of pattern recognition (e.g., (Duda & Hart 1973)). In some situations, using larger partitions reduces the variance of the estimators (see (Kohavi 1995)). In this work, since the number of samples is small, we use LOOCV.

Table 1 lists the accuracy estimates for the different methods applied to the three data sets. As we can see, the clustering approach performs significantly better than the other approaches on the colon cancer data set, but not so on the ovarian data sets. AdaBoost performs better than other methods on the leukemia and overian data sets. We can also see that quadratic SVM does not perform as well as the linear SVM, probably because it overfits the training data. The same phenomenon occurs in Adaboost, where the classifiers are slightly more accurate after 100 iterations than after 10000 iterations.

3.3 ROC Curves

Estimates of classification accuracy give only a partial insight on the performance of a method. In our evaluation, we treated all errors as having equal penalty. In many applications, however, errors have asymmetric weights. For a general discussion of risk and loss considerations in classification see, e.g, (Ripley 1996). To set terminology for our particular case, we distinguish *false positive* errors - normal tissues classified as tumor, and *false negative* errors - tumor tissues classified as normal. In diagnostic applications, false negative errors can be detrimental, while false positives may be tolerated (since additional tests will be performed on the patient).

To deal with asymmetric weights for errors, we introduce the *confidence parameter*, β. In clustering approaches, the modified procedure labels a query sample as tumor if the cluster containing it has at least a fraction β of tumors. In a similar manner, we can introduce confidence parameters for SVM and boosting approaches by changing the threshold margin needed for positive classification.

ROC curves are used to evaluate the "power" of a classification method for different asymmetric weights (see, for example, (Swets 1988)). A ROC curve plots the tradeoff between the two types of errors as the confidence parameter varies. Each point on the two dimensional curve corresponds to a particular value of the confidence parameter. The (x, y) coordinates of a point represent the fractions of negative and positive samples that are classified as positive with this particular confidence parameter. The extreme ends of the curves are the most strict and most permissive confidence values: with the strictest confidence value nothing is classified as positive, putting $(0, 0)$ on the curve; with the most permissive confidence value everything is classified as positive, putting $(1, 1)$ on the curve. The path between these two extremes shows how flexible the procedure is with respect to trading-off error rates. The best case scenario is that the path goes through the point $(0, 1)$. This implies that for some confidence parameter, all positives are classified as positive, and all negatives are classified as negative. That is - the procedure can be made very strict with respect

Table 1: Summary of classification performance of the different methods on the three data sets. The tables shows the precent of samples that were correctly classified, incorrectly classfied, and unclassfied by each method in the LOOCV evaluation. Unsupervised labels for margin based classifier were decided by a fixed threshold on classification margin: in SVM, 0.25, and in Adaboost, 0.05.

Data set	Method	Percent		
		correct	incorrect	unclassified
Colon				
	Clustering	88.7	11.3	0.0
	Nearest Neighbor	80.6	19.4	0.0
	SVM, linear kernel	77.4	12.9	9.7
	SVM, quad. kernel	74.2	14.5	11.3
	Boosting, 100 iter.	72.6	17.7	9.7
	Boosting, 1000 iter.	72.6	17.7	9.7
	Boosting, 10,000 iter.	71.0	19.4	9.7
Ovarian				
	Clustering	42.9	17.9	39.3
	Nearest Neighbor	71.4	28.6	0.0
	SVM, linear kernel	67.9	3.6	28.6
	SVM, quad. kernel	64.3	3.6	32.1
	Boosting, 100 iter.	89.3	10.7	0.0
	Boosting, 1000 iter.	85.7	10.7	3.6
	Boosting, 10,000 iter.	85.7	14.3	0.0
Leukemia				
	Nearest Neighbor	91.6	8.4	0.0
	SVM, linear kernel	93.0	1.4	5.6
	SVM, quad. kernel	94.4	1.4	4.2
	Boosting, 100 iter.	95.8	2.8	1.4
	Boosting, 1000 iter.	95.8	2.8	1.4
	Boosting, 10,000 iter.	95.8	2.8	1.4

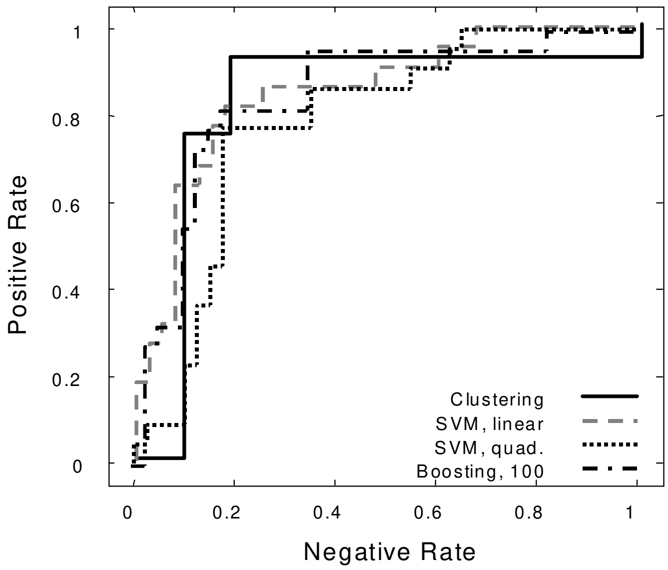


Figure 2: ROC curves for methods applied to the colon cancer data set. The x-axis shows percentage of negative examples classified as positives, and y-axis shows percentage of positive examples classified as positive. Each point along the curve corresponds to the percentages achieved at a particular confidence threshold value by the corresponding classification method. Error estimates are based on LOOCV trials.

to false positive error, with no false negative price to pay. ROC curves with large areas underneath mean that high false positive stringencies can be obtained without much of a false negative price.

In Figure 2 we plot the ROC curves for clustering, SVM and boosting on the colon cancer data set. As we can see, there is no clear domination among the methods. (The only exception is SVM with quadratic kernel that is consistently worse than the other methods.) The clustering procedure is dominant in the region where misclassification errors of both types are roughly of the same importance. However, SVM with linear kernel and boosting are preferable in regions of highly asymmetric error cost (both ends of the spectrum). This may be due to the fact that the matching coefficient score (see Section 2.2), which determines the cluster granularity, treats both types of errors as having equal costs.

4 Gene Selection

It is clear that the expression levels of many of the genes that are measured in our data sets are irrelevant to the distinction between tumor and normal tissues. Taking such genes into account during classification increases the dimensionality of the classification problem, presents computational difficulties, and introduces unnecessary noise to the process. Another issue with a large number of genes is the *interpretability* of the results. If the “signal” that allows our methods to distinguish tumor from normal tissues is encoded in the expression levels of few genes, then we might be able to understand the biological significance of these genes. Moreover, a major goal for diagnostic research is to develop diagnostic procedures based on inexpensive microarrays that have enough probes to detect diseases. Thus, it is crucial to recognize whether a small number of genes can suffice for good classification.

The problem of *feature selection* received a thorough treatment in pattern recognition and machine learning. The gene expression data sets are problematic in that they contain a large number of genes (features) and thus methods that search over subsets of features can be prohibitively expensive. Moreover, these data sets contain only a small number of samples, so the detection of irrelevant genes can suffer from statistical instabilities.

4.1 The TNoM Score

To address these issues, we utilize measures of "relevance" of each gene. In particular, we focus on a quantity we call the *threshold number of misclassification* or *TNoM score* of a gene. The intuition is that an informative gene has quite different values in the two classes (normal and tumor), and thus we should be able to separate these by a threshold value. Formally, we seek the best decision stump for that gene (as defined in Section 2.3.2), and then count the classification errors this decision stump makes on the training examples.

Recall that we describe a decision stump rule by two parameters d, and t. The predicted class is simply sign$(d(x-t))$. The number of errors made by a decision stump is defined as

$$Err(d, t \mid g, l) = \sum_i \mathit{1}\{l_i \neq \text{sign}(d \cdot (x_i[g] - t))\},$$

where $x_i[g]$ is the expression value of gene g in the i'th sample and l_i is the label of the i'th sample.

The TNoM score of a gene is simply defined as:

$$TNoM(g, l) = \min_{d,t} Err(d, t \mid g, l),$$

the number of errors made by the best decision stump. The intuition is that this number reflects the quality of decisions made based on the expression levels of this gene.

Note that the expressiveness of decision stump rules is severely limited. Thus, more expressive queries about the gene's expression level (e.g., whether it is within a particular range) might be more useful for predicting the label of the tissue. However, if we are interested in genes with significant over-expression or under-expression in one of the labels, then such a threshold should separate the two groups.

4.2 Evaluating the Significance of a Gene

An immediate question to ask is whether genes with low TNoM scores are indeed indicative of the classification of expression. In other words, we want to test the statistical significance of the scores of the best scoring genes in our data set. One way to evaluate the significance of such results is to test them against random data. More explicitly: we want to estimate the probability of a gene scoring better than some fixed level s in randomly labeled data. This number is the *p-value* corresponding to the given level s. Genes with very low p-values are very rare in random data and their relevance to the studied phenomenon is therefore likely to have biological, mechanistic or protocol reasons. Genes with low p-values for which the latter two options can be ruled out are interesting subjects for further investigation and are expected to give deeper insight into the studied phenomena.

Let $\{-,+\}^{(n,p)}$ denote all vectors with n $'-'$ entries and p $'+'$ entries (the normal/cancer semantic is one possible interpretation). Let u be a vector of labels. Also let g be a vector of gene expression values. The TNoM score is a function that takes g and u and returns the score of g with respect to labeling u.

We want to compute the p-value of a score on a particular gene. We assume that the vector of gene expression values g is fixed, and consider random label assignments. Let $U_{n,p}$ be a random vector drawn uniformly over $\{-,+\}^{(n,p)}$. The p-value of a score s is then

$$pVal(s : g, n, p) = Prob(\,TNoM(g, U_{n,p}) \leq s). \tag{3}$$

We assume, without loss of generality, that the values in g appear in ascending order. Note that the $TNoM$ score is insensitive to the actual distance between consecutive expression values of the gene. Thus, when we examine p-values, we do not need to examine the specifics of g. Therefor, we can use the shorthands $pVal(s : n, p)$ and $TNoM(u)$.

The combinatorial character of TNoM makes it amenable to rigorous calculations. We now describe a recursive procedure that computes the exact distribution of TNoM scores in $\{-,+\}^{(n,p)}$. We start by defining the two one-sided TNoM scores:

Definition 4.1 *Let $v \in \{-,+\}^{(n,p)}$. Define:*

$$s^{-+}(v) = \min_{i=0\ldots n+p} \left(\begin{array}{c} \text{\# of } +\text{s in } v_1, \ldots, v_i \\ + \\ \text{\# of } -\text{s in } v_{i+1}, \ldots, v_{n+p} \end{array} \right),$$

and symmetrically:

$$s^{+-}(v) = \min_{i=0\ldots n+p} \left(\begin{array}{c} \text{\# of } -\text{s in } v_1, \ldots, v_i \\ + \\ \text{\# of } +\text{s in } v_{i+1}, \ldots, v_{n+p} \end{array} \right).$$

Intuitively, $s^{-+}(v)$ is the score of the labeling v when we only examine decision stumps in which values above the threshold are labeled $+$ and values below the threshold are labeled $-$. (That is, the d coefficient is positive). Similarly, $s^{+-}(v)$ is the score of the labeling when we impose the symmetric constraint.

The following simple proposition is the basis of the recursive step:

Proposition 4.2 *Let $v \in \{-,+\}^{(n,p)}$. There are two cases:*

1. *if $v = u+$, for $u \in \{-,+\}^{(n,(p-1))}$, then*

$$s^{+-}(v) = \begin{cases} s^{+-}(u) + 1 & \text{if } s^{+-}(u) < n \\ s^{+-}(u) & \text{if } s^{+-}(u) = n \end{cases}$$

   and

$$s^{-+}(v) = s^{-+}(u),$$

2. *if $v = u-$, for $u \in \{-,+\}^{((n-1),p)}$, then*

$$s^{+-}(v) = s^{+-}(u),$$

   and

$$s^{-+}(v) = \begin{cases} s^{-+}(u) + 1 & \text{if } s^{-+}(u) < p \\ s^{-+}(u) & \text{if } s^{-+}(u) = p \end{cases}$$

Suppose we are now interested in computing all TNoM score distributions for the spaces $\{-,+\}^{(n,p)}$ where n ranges from 0 to ν and p ranges from 0 to π. We define an array T as follows

$$T(n,p,s,t) = \left| \left\{ v \in \{-,+\}^{(n,p)} : \begin{array}{c} s^{-+}(v) = s \\ \text{and} \\ s^{+-}(v) = t \end{array} \right\} \right|$$

Contributions to $T(n,p,s,t)$ come from vectors in $\{-,+\}^{((n-1),p)}$ by concatenating a $-$ and from vectors in $\{-,+\}^{(n,(p-1))}$ by concatenating a $+$. Proposition 4.2 indicates the size of each such contribution in the various cases and we obtain the following recursion formula:

$$\begin{aligned} T(n,p,s,t) &= T(n,p-1,s,t-1) + \\ &\quad T(n-1,p,s-1,t) + \\ &\quad \mathbf{1}[t=n] \cdot T(n,p-1,s,n) + \\ &\quad \mathbf{1}[s=p] \cdot T(n-1,p,p,t), \end{aligned} \tag{4}$$

where $\mathbf{1}[t=n]$ is 1 iff $t=n$ and 0 otherwise.

Initial conditions for this recursive calculation are trivially set. To obtain the explicit distribution and thus the p-values we use the following formula:

$$Prob\,(TNoM(U_{n,p}) = s) = \frac{T(n,p,s,s) + 2 \cdot \sum_{t>s} T(n,p,s,t)}{\binom{n+p}{p}}.$$

4.3 Informative Genes in Cancer Data

Consider a set of actual labeled gene expression data, such as the ones we described above. It is beneficial to give some quantitative score to the abundance of highly informative genes, with respect to the given labeling. Figure 3 depicts a comparison between the expected number of genes scoring better than a given threshold and the actual number found in the data. As we can see, the number of highly informative genes is well above the expected number according to the null-hypothesis.

To evaluate the biological meaning of the high scoring genes we have ordered the genes in the data sets, according to their TNoM scores, and examined the genes at the top of the list (those with better TNoM scores). Appendix A lists the high-scoring genes in the colon and leukemia data sets.

Among the top scoring genes ($TNoM \leq 14$) in the colon cancer data set there are a number of genes that are interesting from the perspective of a potential involvement in tumorigenesis including, for example, genes involved in cell cycle regulation and angiogenesis. There were also genes, for example (D63874) HMG-1 (human) and (T55840) tumor-associated antigen L6 (human), that have previously been found to have a particular association with colorectal carcinomas (Schiedeck et al. 1998, Xiang et al. 1997).

Among the top scoring 137 clones in the ovarian cancer data, there are 85 clones that match 8 cancer related genes (potential markers or expressed in cancer cells) and one gene that is related to increased metabolic rate (mitochondrial gene). The 8 genes are keratin 18 (breast cancer), pyruvate kinase muscle 2 (hepatoma), thymopoietin (cell proliferation), HE4 (ovarian cancer), SLPI (many

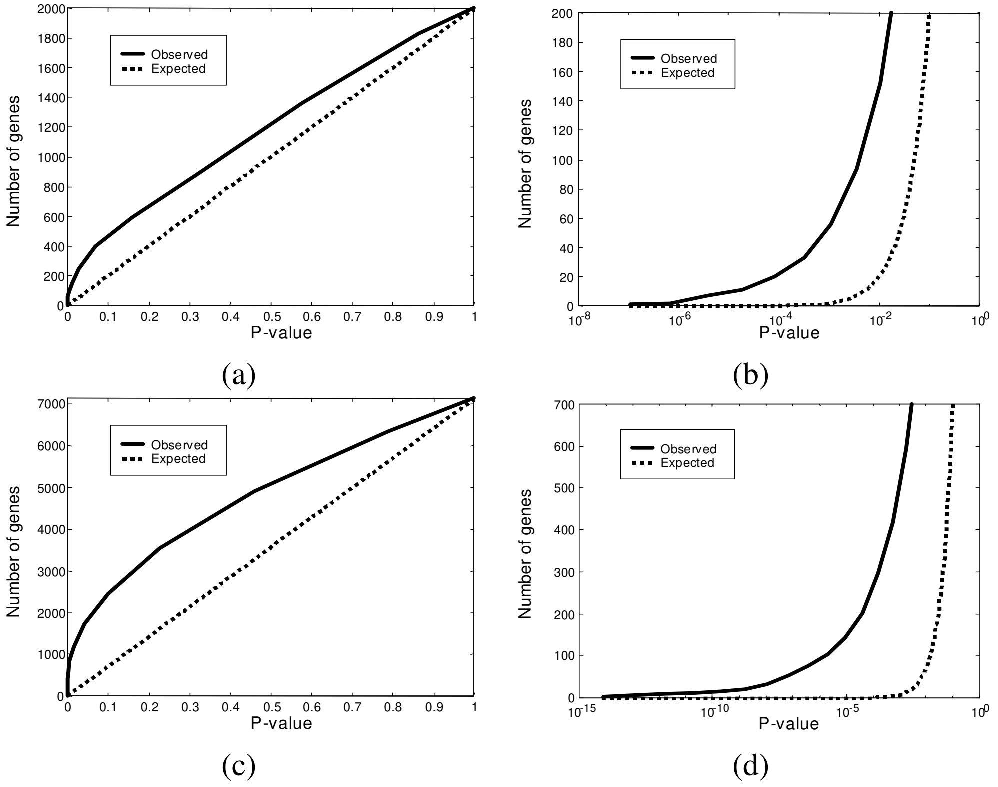


Figure 3: Comparison of the number of significant genes in actual data sets to the expected number under the null-hypothesis (random labels). The x-axis denotes p-value and the y-axis the number of genes. The expected number of genes with score $TNoM$ better than a given s is $Prob(TNoM(U_{n,p}) \leq s) \cdot (\text{\# of genes})$. Graphs (a) and (b) show results from the Colon data set. Graphs (c) and (d) show results from the Leukemia data set. The graphs on the left, (a) and (c), show the whole significance range, and the graphs on the right, (b) and (d), show the tail of the distribution (p-values are in log-scale).

different cancers, among them lung, breast, oropharyngeal, bladder, endometrial, ovarian and colorectal carcinoma), ferritin H (ovarian cancer), collagen 1A1 (ovarian cancer, osteosarcoma, cervical carcinoma), and GAPDH (cancers of lung, cervix and prostate). In addition, 2 clones with no homology to a known gene are found in this selection. Given the high number of cancer related genes in the top 137, it is likely that these novel genes exhibit a similar cancer-related behavior. We conducted expression validation for GAPDH, SLPI, HE4 and keratin 18 which confirmed the elevated expression in some ovarian carcinomas compared to normal ovarian tissues.

4.4 Classifying with Selected Subsets

When using gene selection, we need to pre-process the training data to select genes. Then, the classification procedure is applied using the training data restricted to the subset of selected genes. The gene selection stage is given a parameter k, which determines the largest error-score allowed. It then selects all genes that have a smaller or equal error score on the training data. Alternatively, a p-value approach can be taken: all genes with scores which are very rare in random data are selected.

To evaluate performance with gene selection, we have to be careful to jointly evaluate both stages of the process: gene selection and classification. Thus, in each cross-validation trial, gene selection is applied based on the training examples in that trial. Note, that since the training examples are different in different cross validation trials, we expect the number of selected genes to depend on the trial.

Figure 4 describes the performance of some of the methods we discussed above when we vary the stringency of the selection process.

In the colon data set, gene selection leads to mixed results. Some methods, such as clustering, perform slightly worse with fewer genes, while others, such as SVM, perform better with smaller set of genes. On the other hand, in the ovarian data set, gene selection leads to impressive improvement in all methods. All methods perform well in the region between threshold 3 (avg. 173 clones) to 6 (avg. 4375 clones). Note that both Boosting and SVM perform well even with fewer clones. In the leukemia data set, gene selection slightly improved the performance of AdaBoost (which performed well with all the genes), and significantly improved the performance of other methods (between threshold of 11 to 13 $TNoM$ score).

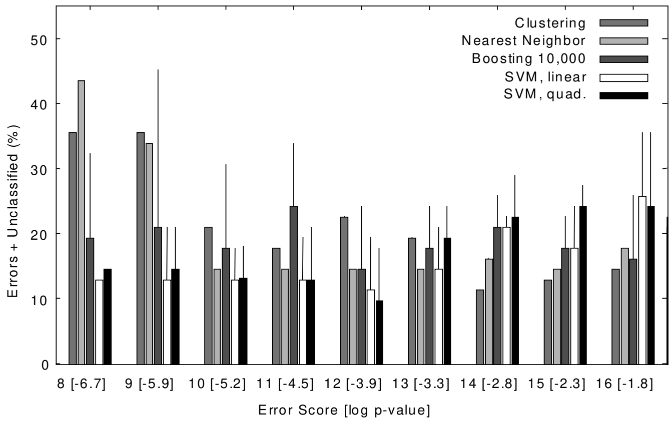
Figure 5 shows ROC curves for Clustering approach, Boosting, and quadratic SVM with threshold of 3 (linear SVM has similar curve to quadratic SVM, and thus was not plotted). As we can see, although all methods have roughly the same accuracy with this subset of genes, their ROC profile is strikingly different. These curves clearly show that the Clustering approach makes false positive errors, while all the other approaches make false negative errors.

It is instructive to compare the TNoM scores of genes to other methods for selecting genes. In particular, we note that the AdaBoost procedure is effectively a gene selection method. In each iteration the AdaBoost procedure selects a stump classifier that examines the expression value of a single gene. Thus, we can evaluate the importance of a gene in the AdaBoost classification by the weight assigned to decision stumps that queries the value of that gene. Figure 6 shows a comparison of gene AdaBoost weights to TNoM scores. As we can see, the highest weight genes have low TNoM scores. In the leukemia data set we also see that there is a correlation between gene weight and the TNoM score. Note that in this data set, AdaBoost is very effective without additional gene selection. On the other hand, in the colon data set, AdaBoost's performance is improved when we remove genes with high TNoM score. We note that in general the AdaBoost procedure does not use all of the genes with low TNoM score. This suggests that there is a significant overlap in the information that these genes convey about the classification.

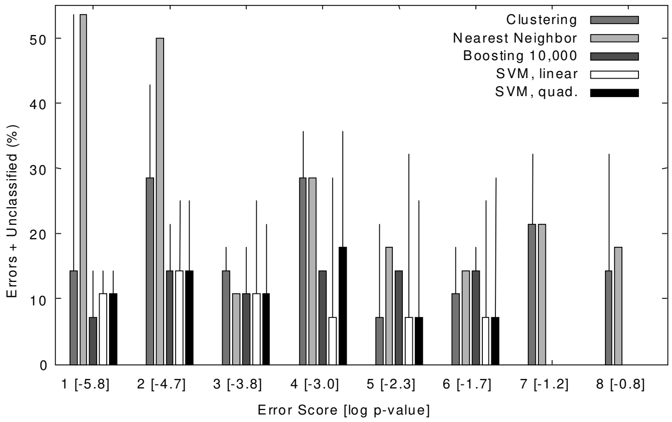
5 Sample Contamination

Cancer classification based on array-based gene expression profiling may be complicated by the fact that clinical samples, e.g. tumor vs. normal, will likely contain a mixture of different cell types. In addition, the genomic instability inherent in tumor samples may lead to a large degree of random fluctuations in gene expression patterns. Although both the biological and genetic variability in tumor samples have the potential to lead to confusing and difficult to interpret expression profiles, gene expression profiling does allow us to efficiently distinguish tumor and normal samples, as

Colon



Ovarian



Leukemia

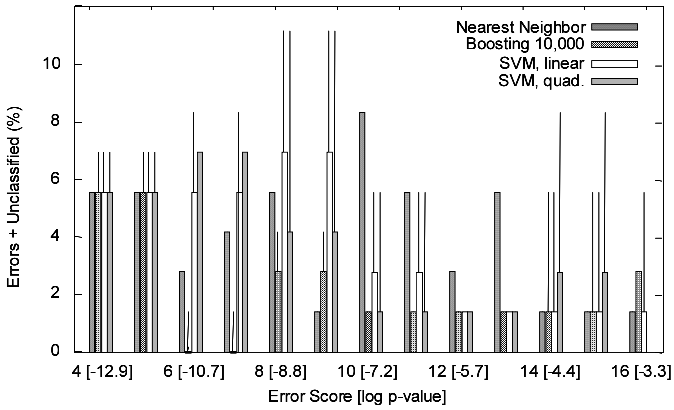


Figure 4: Classification performance, as it depends on the threshold used for selecting genes. The x-axis shows the TNoM score threshold used and the base 10 logarithm of the associated p-value. The results are based on performing LOOCV for the whole process of selection and classification, as explained in the text. For each method, the solid bar represents the fraction of the training data that was mis-classified. The thin line extensions represent the fraction which was unclassified.

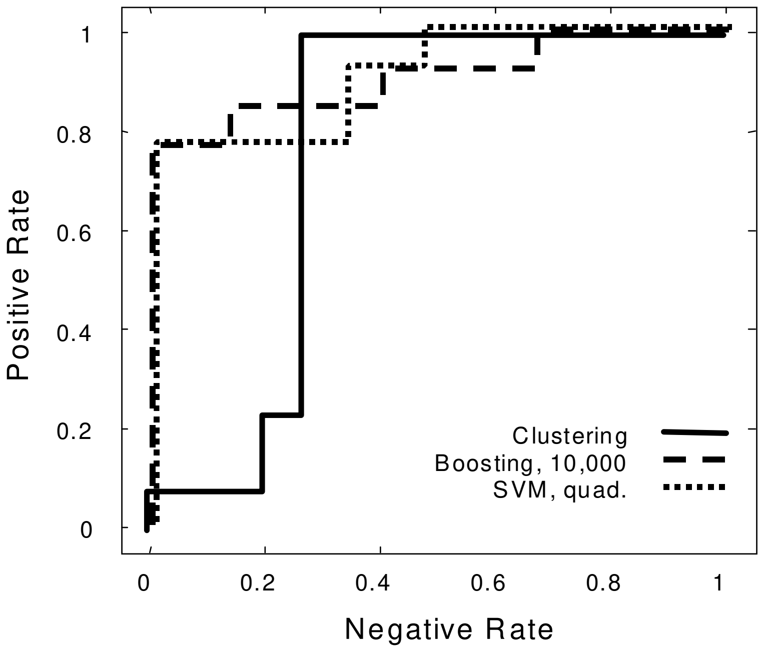


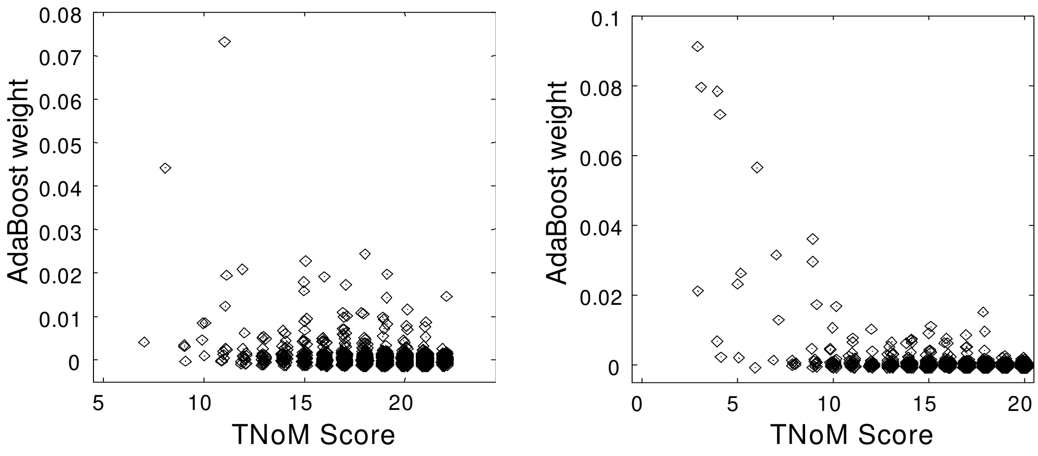
Figure 5: ROC curves for three methods that are applied to the ovarian data set with TNoM score threshold set to 3

we have seen in the previous sections. However, the presence of different cell types within and between samples could lead to identification of genes that strongly affect cluster formation but which may have little to do with the process being studied, in this case tumorigenesis. For example, in the case of the colon cancer data set presented above, a large number of muscle-specific genes were identified as being characteristic of normal colon samples both in our clustering results and in the results of Alon et al. (1999). This is most likely due to a higher degree of smooth muscle contamination in the normal versus tumor samples.

This raises the concern that our classification may be biased by the presence of muscle specific genes. To test this hypothesis, we attempted to construct data sets that avoid genes that are suspected in introducing bias. We listed the top 200 error-score ranking genes in the colon cancer data set, and identified muscle-specific genes. These include (J02854) myosin regulatory light chain 2, smooth muscle isoform (human); (T60155) actin, aortic smooth muscle (human); and (X12369) tropomyosin alpha chain, smooth muscle (human)that are designated as smooth muscle-specific by Alon et al.'s analysis, and (M63391) desmin (human), complete cds; (D31885) muscle-specific EST (human); and (X7429) alpha 7B integrin (human) which are suspected to be expressed in smooth muscle based on literature searches.

An additional form of "contamination" is due to the high metabolic rate of the tumors. This results in high expression values for ribosomal genes. Although such high expression levels can be indicative of tumors, such a finding does not necessarily provide novel biological insight into the process, nor does it provide a diagnostic tool since ribosomal activity is present in virtually all tissues. Thus, we also identified ribosomal genes in the top 200 scoring genes.

Figure 7 shows the performance of the clustering approach on three data sets: the full 2000 gene data set, a data set without muscle specific genes, and a data set without both muscle specific



Colon Leukemia

Figure 6: Comparison of the weight assigned to genes in the AdaBoost classification (without gene selection) and the TNoM score. Each point in the scatter plot corresponds to a gene. The x-axis denotes the TNoM score of the gene, and the y-axis the weight associated with all the decision stumps that query the gene's expression value in the classifier learned by AdaBoost.

and ribosomal genes. As the learning curves show, the removal of genes affects the results only in cases using the smallest sets of genes. From error score threshold of 10 (avg. 9.1 genes) and higher, there is no significant change in performance for the procedure. Thus, although muscle specific genes can be highly indicative, the classification procedure performs well even without relying on these genes.

Although the muscle contamination did not necessarily alter the ability of this gene set to be used to classify tumor vs. normal samples in this case, it will continue to be important to account for possible affects of tissue contamination on clustering and classification results. Experimental designs that include gene expression profiles of tissue and/or cell culture samples representative of types of tissue contaminants known to be isolated along with different types of tumor samples (for example see Perou et al. (1999)), can be utilized to help distinguish contaminant gene expression profiles from those actually associated with specific types of tumor cells.

6 Conclusions

In this paper we examined the question of tissue classification based on expression data. Our contribution is four-fold. First, we introduced a new cluster-based approach for classification. This approach builds on clustering algorithms that are suitable for gene expression data. Second, we performed rigorous evaluation of this method, and of known methods from the machine learning literature. These include large margin classification methods (SVM and AdaBoost) and the nearest-neighbor method. Third, we highlighted the issue of sample contamination and estimated the sensitivity of our approach to sample variability. Differences in tissue biopsies could theoretically affect the quality of any given classification method. Studying this issue, we observed no

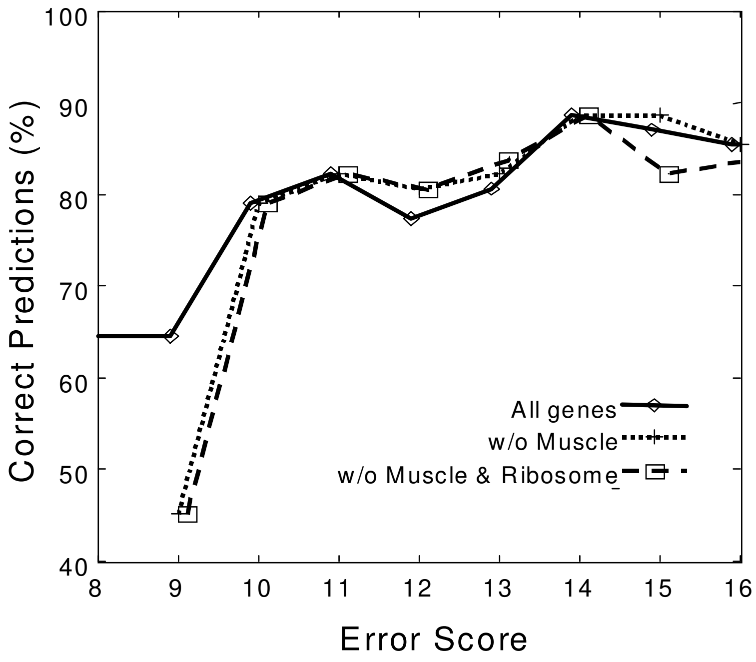


Figure 7: Curves showing the predictive performance of clustering methods in the original Alon et al. data set, and data sets where muscle specific, and ribosomal genes were removed. All estimates are based on LOOCV evaluation. These results show that even without the obvious contaminations, our methods are successful in reliably predicting tissue type.

significant contaminating tissue bias in the colon cancer data set. Finally, we investigated the issue of gene selection in expression data. As our results for the ovarian data set show, a large number of clones can have a negative impact on predictive performance. We showed that a fairly simple selection procedure can lead to significant improvements in prediction accuracy. In addition, we derived an efficient dynamic programming method for computing *exact* p-values for a gene's TNoM score.

The work reported here is closely related to two recent papers. First, Golub et al. (1999) (see also (Slonim et al. 2000)) examined a scoring rule to select informative genes and performed LOOCV experiments to test a voting based classification approach. Although their score for gene selection and their classification method are different than ours, their main conclusions are quite similar in that they get good classification accuracy with relatively small number of genes. Our results on the same data set (leukemia) are comparable or better. These results emphasize the conclusion that the two leukemia phenotypes (ALL and AML) are well seperated in expression data.

Second, Brown et al. (1999) use support vector machines in the context of gene expression data. In contrast to our approach, they attempt to classify the genes rather then samples. Thus, they deal with the dual classification problem. The characteristics of their classification problem are quite different: many examples (i.e., thousands of genes), and few attributes (i.e., expression in different samples). We note that some of the approaches we used in this work (e.g., clustering based classification) might be applicable to this dual classification problem as well.

As noted above, the gene selection process we explored in this paper is quite simplistic. In particular, it was based on scoring single genes for relevance. Thus, the process might select several genes that convey the same information, and might ignore genes that add independent information. We are currently studying more direct approaches to the selection of informative *sets* of genes. Identifying sets of genes that give rise to efficient learned classifiers might reveal previously unknown disease related genes and guide further biological research.

Acknowledgments

The authors are grateful to Yoram Singer for help with SVM.

References

Alon, U., Barkai, N., Notterman, D., Gish, K., Ybarra, S., Mack, D. & Levine, A. J. (1999), 'Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays', *Proc. Nat. Acad. Sci. USA* **96**, 6745–6750.

Ben-Dor, A., Shamir, R. & Yakhini, Z. (1999), 'Clustering gene expression patterns', *Journal of Computational Biology* **6**, 281–297.

Bishop, C. M. (1995), *Neural Networks for Pattern Recognition*, Oxford University Press, Oxford, U.K.

Brown, M., Grundy, W., Lin, D., Cristianini, N., Sugnet, C., Furey, T., Jr., M. A. & Haussler, D. (1999), Knowledge-based analysis of microarray gene expression data using support vector machines, Technical Report UCSC-CRL-99-09, U. C. Santa Cruz.

Burges, C. J. C. (1998), 'A tutorial on Support Vector Machines for pattern recognition', *Data Mining and Knowledge Discovery* **2**, 121–167.

Chu, S., DeRisi, J., Eisen, M., Mullholland, J., Botstein, D., Brown, P. & Herskowitz, I. (1998), 'The transcriptional program of sporulation in budding yeast', *Science* **282**, 699–705.

Clarke, P. A., George, M., Cunningham, D., Swift, I. & Workman, P. (1999), Analysis of tumor gene expression following chemotherapeutic treatment of patients with bowel cancer, *in* 'Proc. Nature Genetics Microarray Meeting 99', Scottsdale, Arizona, p. 39.

Cortes, C. & Vapnik, V. (1995), 'Support vector machines', *Machine Learning* **20**, 273–297.

DeRisi., J., Iyer, V. & Brown, P. (1997), 'Exploring the metabolic and genetic control of gene expression on a genomic scale', *Science* **282**, 699–705.

Duda, R. O. & Hart, P. E. (1973), *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York.

Eisen, M., Spellman, P., Brown, P. & Botstein, D. (1998), 'Cluster analysis and display of genome-wide expression patterns', *Proc. Nat. Acad. Sci. USA* **95**, 14863–14868.

Everitt, B. (1993), *Cluster Analysis*, third edn, Edward Arnold, London.

Freund, Y. & Schapire, R. E. (1997), 'A decision-theoretic generalization of on-line learning and an application to boosting', *J. Computer and System Sciences* **55**, 119–139.

Golub, T., Slonim, D., Tamayo, P., Huard, C., M. Caasenbeek, J. M., Coller, H., Loh, M., Downing, J., Caligiuri, M., Bloomfield, C. & Lander, E. (1999), 'Molecular classification of cancer: class discovery and class prediction by gene expression monitoring', *Science* **286**, 531–537.

Iyer, V., Eisen, M., Ross, D., Schuler, G., Moore, T., Lee, J., Trent, J., Staudt, L., Hudson, J., Boguski, M., Lashkari, D., Shalon, D., Botstein, D. & Brown, P. (1999), 'The transcriptional program in the response of human fibroblasts to serum', *Science* **283**, 83–87.

Khan, J., Simon, R., Bittner, M., Chen, Y., Leighton, S. B., Pohida, T., Smith, P. D., Jiang, Y., Gooden, G. C., Trent, J. M. & Meltzer, P. S. (1998), 'Gene expression profiling of *Alveolar rhabdomyosarcoma* with cDNA microarrays', *Cancer Reasearch* .

Kohavi, R. (1995), A study of cross-validation and bootstrap for accuracy estimation and model selection, *in* 'Proc. Fourteenth International Joint Conference on Artificial Intelligence (IJCAI '95)', Morgan Kaufmann, San Francisco, Calif., pp. 1137–1143.

Lockhart, D. J., Dong, H., Byrne, M. C., Follettie, M. T., Gallo, M. V., Chee, M. S., Mittmann, M., Want, C., Kobayashi, M., Horton, H. & Brown, E. L. (1996), 'DNA expression monitoring by hybridization of high density oligonucleotide arrays', *Nature Biotechnology* **14**, 1675–1680.

Mason, L., Bartlett, P. & Baxter, J. (1999), Direct optimization of margins improves generalization in combined classifiers, *in* 'Advances in Neural Information Processing Systems 11', MIT Press, Cambridge, Mass.

Perou, C. M., Jeffrey, S. S., v de Rijn, M., Rees, C. A., Eisen, M. B., Ross, D. T., Pergamenschikov, A., Williams, C. F., Zhu, S. X., Lee, J. C. F., Lashkari, D., Shalon, D., Brown, P. O. & D, B. (1999), 'Distinctive gene expression patterns in human mammary epithelial cells and breast cancers', *Proc. Nat. Acad. Sci. USA* **96**, 9212–9217.

Ripley, B. D. (1996), *Pattern Recognition and Neural Networks*, Cambridge University Press.

Schapire, R. E. (1990), 'The strength of weak learnability', *Machine Learning* **5**, 197–227.

Schapire, R. E., Freund, Y., Bartlett, P. & Lee, W. S. (1998), 'Boosting the margin: A new explanation for the effectiveness of voting methods', *Annals of Statistics* **26**, 1651–1686.

Schiedeck, T., Christoph, S., Duchrow, M. & Bruch, H. (1998), 'Detection of hl6-mRNA: new possibilities in serologic tumor diagnosis of colorectal carcinomas', *Zentralbl Chir* **123**(2), 159–162.

Schummer, M., NG, W., Bumgarner, R., Nelson, P., Schummer, B., Hassell, L., Baldwin., L. R., Karlan, B. & Hood, L. (1999), 'Comperative hybridization of an array of 21,500 ovarian cDNAs for the discovery of genes overexpressed in ovarian carcinomas', *Gene* **238**, 375–385.

Slonim, D. K., Tamayo, P., Mesirov, J. P., Golub, T. R. & Lander, E. S. (2000), Class prediction and discovery using gene expression data, *in* ‘Fourth Annual International Conference on Computational Molecular Biology’, pp. 263–272.

Swets, J. (1988), ‘Measuring the accuracy of diagnostic systems’, *Science* **240**, 1285–1293.

Vapnik, V. (1999), *Statistical Learning Theory*, John Wiley & Sons, New York.

Wen, X., Furhmann, S., Micheals, G. S., Carr, D. B., Smith, S., Barker, J. L. & Somogyi, R. (1998), ‘Large-scale temporal gene expression mapping of central nervous system development’, *Proc. Nat. Acad. Sci. USA* **95**, 334–339.

Xiang, Y., Wang, D., Tanaka, M., Suzuki, M., Kiyokawa, E., Igarashi, H., Naito, Y., Shen, Q. & Sugimura, H. (1997), ‘Expression of high-mobility group-1 mRNA in human gastrointestinal adenocarcinoma and corresponding non-cancerous mucosa’, *Int J. Cancer* **74**(1), 1–6.

A Top Scoring Genes

A.1 Colon Cancer Data Set

TNoM	Gene	Description
7	M63391	Human desmin gene, complete cds.
8	M26383	Human monocyte-derived neutrophil-activating protein (MONAP) mRNA, complete cds.
9	R87126	197371 MYOSIN HEAVY CHAIN, NONMUSCLE (Gallus gallus)
9	M76378	Human cysteine-rich protein (CRP) gene, exons 5 and 6.
9	M76378	Human cysteine-rich protein (CRP) gene, exons 5 and 6.
9	M22382	MITOCHONDRIAL MATRIX PROTEIN P1 PRECURSOR (HUMAN);.
9	J05032	Human aspartyl-tRNA synthetase alpha-2 subunit mRNA, complete cds.
10	M76378	Human cysteine-rich protein (CRP) gene, exons 5 and 6.
10	R36977	26045 P03001 TRANSCRIPTION FACTOR IIIA ;.
10	H40095	175181 MACROPHAGE MIGRATION INHIBITORY FACTOR (HUMAN);.
10	J02854	MYOSIN REGULATORY LIGHT CHAIN 2, SMOOTH MUSCLE ISOFORM (HUMAN);contains element TAR1 repetitive element ;.
11	H08393	45395 COLLAGEN ALPHA 2(XI) CHAIN (Homo sapiens)
11	X12671	Human gene for heterogeneous nuclear ribonucleoprotein (hnRNP) core protein A1.
11	T96873	121343 HYPOTHETICAL PROTEIN IN TRPE)
11	X63629	H.sapiens mRNA for p cadherin.
11	U25138	Human MaxiK potassium channel beta subunit mRNA, complete cds.
11	T71025	84103 Human (HUMAN);.
11	T92451	118219 TROPOMYOSIN, FIBROBLAST AND EPITHELIAL MUSCLE-TYPE (HUMAN);.
11	U09564	Human serine kinase mRNA, complete cds.
11	R64115	139618 ADENOSYLHOMOCYSTEINASE (Homo sapiens)
12	R42501	29607 INOSINE-5'-MONOPHOSPHATE DEHYDROGENASE 2 (HUMAN);.
12	T86473	114645 NUCLEOSIDE DIPHOSPHATE KINASE A (HUMAN);.
12	T47377	71035 S-100P PROTEIN (HUMAN).
12	X14958	Human hmgI mRNA for high mobility group protein Y.
12	D31885	Human mRNA (KIAA0069) for ORF (novel proetin), partial cds.
12	M36634	Human vasoactive intestinal peptide (VIP) mRNA, complete cds.
12	X86693	H.sapiens mRNA for hevin like protein.
12	T60778	76539 MATRIX GLA-PROTEIN PRECURSOR (Rattus norvegicus)
12	U29092	Human ubiquitin conjugating enzyme mRNA, complete cds.
12	X54942	H.sapiens ckshs2 mRNA for Cks1 protein homologue.
12	T60155	81422 ACTIN, AORTIC SMOOTH MUSCLE (HUMAN);.
12	M36981	Human putative NDP kinase (nm23-H2S) mRNA, complete cds.
12	T79152	113545 60S RIBOSOMAL PROTEIN L19 (HUMAN);.
13	X53586	Human mRNA for integrin alpha 6.
13	H43887	183264 COMPLEMENT FACTOR D PRECURSOR (Homo sapiens)
13	X56597	Human humFib mRNA for fibrillarin.
13	H77597	214162 H.sapiens mRNA for metallothionein (HUMAN);.
13	M26697	Human nucleolar protein (B23) mRNA, complete cds.
13	X70326	H.sapiens MacMarcks mRNA.
13	R08183	127228 Q04984 10 KD HEAT SHOCK PROTEIN, MITOCHONDRIAL ;.
13	R52081	40295 TRANSCRIPTIONAL ACTIVATOR GCN5 (Saccharomyces cerevisiae)
13	T95018	120032 40S RIBOSOMAL PROTEIN S18 (Homo sapiens)
13	X12466	Human mRNA for snRNP E protein.
13	Z49269	H.sapiens gene for chemokine HCC-1.
13	X62048	H.sapiens Wee1 hu gene.
13	T61609	78081 LAMININ RECEPTOR (HUMAN);.
13	T67077	66563 SODIUM/POTASSIUM-TRANSPORTING ATPASE GAMMA CHAIN (Ovis aries)
13	U19969	Human two-handed zinc finger protein ZEB mRNA, partial cds.
13	X15183	Human mRNA for 90-kDa heat-shock protein.
13	T57633	75467 40S RIBOSOMAL PROTEIN S8 (HUMAN).
13	M91463	Human glucose transporter (GLUT4) gene, complete cds.
13	D29808	Human mRNA for T-cell acute lymphoblastic leukemia associated antigen 1 (TALLA-1), complete cds.
13	T51023	75127 HEAT SHOCK PROTEIN HSP 90-BETA (HUMAN).
13	H87135	252431 IMMEDIATE-EARLY PROTEIN IE180 (Pseudorabies virus)
13	T83368	116679 MEMBRANE COFACTOR PROTEIN PRECURSOR (Homo sapiens)
13	T51529	72384 ELONGATION FACTOR 1-DELTA (Artemia salina)

TNoM	Gene	Description
14	U30825	Human splicing factor SRp30c mRNA, complete cds.
14	Z50753	H.sapiens mRNA for GCAP-II/uroguanylin precursor.
14	U32519	Human GAP SH3 binding protein mRNA, complete cds.
14	R84411	194660 SMALL NUCLEAR RIBONUCLEOPROTEIN ASSOCIATED PROTEINS B AND B' (HUMAN);.
14	H40560	175410 THIOREDOXIN (HUMAN);.
14	T62947	79366 60S RIBOSOMAL PROTEIN L24 (Arabidopsis thaliana)
14	T51571	72250 P24480 CALGIZZARIN.
14	X55715	Human Hums3 mRNA for 40S ribosomal protein s3.
14	T52185	71940 P17074 40S RIBOSOMAL PROTEIN.
14	D63874	Human mRNA for HMG-1.
14	Z49269	H.sapiens gene for chemokine HCC-1.
14	U17899	Human chloride channel regulatory protein mRNA, complete cds.
14	L41559	Homo sapiens pterin-4a-carbinolamine dehydratase (PCBD) mRNA, complete cds.
14	H64489	238846 LEUKOCYTE ANTIGEN CD37 (Homo sapiens)
14	L08069	Human heat shock protein, E. coli DnaJ homologue mRNA, complete cds.
14	H89087	253224 SPLICING FACTOR SC35 (Homo sapiens)
14	R75843	143567 TRANSLATIONAL INITIATION FACTOR 2 GAMMA SUBUNIT (Homo sapiens)
14	T40454	60221 ANTIGENIC SURFACE DETERMINANT PROTEIN OA3 PRECURSOR (Homo sapiens)
14	H06524	44386 GELSOLIN PRECURSOR, PLASMA (HUMAN);.
14	T57630	75459 S34195 RIBOSOMAL PROTEIN L3 -.
14	D00596	Human thymidylate syntase (EC 2.1.1.45) gene, complete cds.
14	U26312	Human heterochromatin protein HP1Hs-gamma mRNA, partial cds.
14	L05144	PHOSPHOENOLPYRUVATE CARBOXYKINASE, CYTOSOLIC (HUMAN);contains Alu repetitive element;contains element PTR5 repetitive element ;.
14	M80815	H.sapiens a-L-fucosidase gene, exon 7 and 8, and complete cds.
14	X74295	H.sapiens mRNA for alpha 7B integrin.
14	T86749	114310 Human (clone PSK-J3) cyclin-dependent protein kinase mRNA, complete cds.,.
14	M64110	Human caldesmon mRNA, complete cds.
14	X70944	H.sapiens mRNA for PTB-associated splicing factor.
14	X74262	H.sapiens RbAp48 mRNA encoding retinoblastoma binding protein.
14	R78934	146232 ENDOTHELIAL ACTIN-BINDING PROTEIN (Homo sapiens)
14	H11719	47679 MONOCYTE DIFFERENTIATION ANTIGEN CD14 PRECURSOR (HUMAN);.
14	H55916	204131 PEPTIDYL-PROLYL CIS-TRANS ISOMERASE, MITOCHONDRIAL PRECURSOR (HUMAN);.
14	D42047	Human mRNA (KIAA0089) for ORF (mouse glycerophosphate dehydrogenase-related), partial cds.
14	L25941	Homo sapiens integral nuclear envelope inner membrane protein (LBR) gene, complete cds.
14	H20819	51442 26S PROTEASE REGULATORY SUBUNIT 6 (Homo sapiens)
14	X12496	Human mRNA for erythrocyte membrane sialoglycoprotein beta (glycophorin C).
14	X13482	U2 SMALL NUCLEAR RIBONUCLEOPROTEIN A' (HUMAN);contains MER22 repetitive element ;.

A.2 Leukemia Data Set

TNoM	Gene	Description
3	M23197	CD33 CD33 antigen (differentiation antigen)
3	X95735	Zyxin
3	M27891	CST3 Cystatin C (amyloid angiopathy and cerebral hemorrhage)
4	U46499	GLUTATHIONE S-TRANSFERASE, MICROSOMAL
4	D88422	CYSTATIN A
4	M84526	DF D component of complement (adipsin)
4	M31523	TCF3 Transcription factor 3 (E2A immunoglobulin enhancer binding factors E12/E47)
5	L09209	APLP2 Amyloid beta (A4) precursor-like protein 2
5	M11722	Terminal transferase mRNA
5	M83652	PFC Properdin P factor, complement
6	M92287	CCND3 Cyclin D3
6	X62320	GRN Granulin
7	X62654	ME491 gene extracted from H.sapiens gene for Me491/CD63 antigen
7	J05243	SPTAN1 Spectrin, alpha, non-erythrocytic 1 (alpha-fodrin)
7	M96326	Azurocidin gene
8	X59417	PROTEASOME IOTA CHAIN
8	M31211	MYL1 Myosin light chain (alkali)
8	M63138	CTSD Cathepsin D (lysosomal aspartyl protease)
8	M55150	FAH Fumarylacetoacetate
8	X52056	SPI1 Spleen focus forming virus (SFFV) proviral integration oncogene spi1
9	X61587	ARHG Ras homolog gene family, member G (rho G)
9	X17042	PRG1 Proteoglycan 1, secretory granule
9	M19507	MPO Myeloperoxidase
9	U50136	Leukotriene C4 synthase (LTC4S) gene
9	M63379	CLU Clusterin (complement lysis inhibitor; testosterone-repressed prostate message 2; apolipoprotein J)
9	M16038	LYN V-yes-1 Yamaguchi sarcoma viral related oncogene homolog
9	Z15115	TOP2B Topoisomerase (DNA) II beta (180kD)
9	M22960	PPGB Protective protein for beta-galactosidase (galactosialidosis)
9	HG1612-HT1612	Macmarcks
9	M31303	Oncoprotein 18 (Op18) gene
9	M83667	NF-IL6-beta protein mRNA
9	HG3494-HT3688	Nuclear Factor Nf-Il6
10	D88270	GB DEF = (lambda) DNA for immunoglobin light chain
10	U05259	MB-1 gene
10	D14664	KIAA0022 gene
10	M93056	LEUKOCYTE ELASTASE INHIBITOR
10	X90858	Uridine phosphorylase
10	X85116	Epb72 gene exon 1
10	X16546	RNS2 Ribonuclease 2 (eosinophil-derived neurotoxin; EDN)
10	M11147	FTL Ferritin, light polypeptide
10	M14636	PYGL Glycogen phosphorylase L (liver form)
10	M62762	ATP6C Vacuolar H+ ATPase proton channel subunit
10	U05572	MANB Mannosidase alpha-B (lysosomal)
10	X64072	SELL Leukocyte adhesion protein beta subunit
10	M19508	MPO from Human myeloperoxidase gene, exons 1-4./ntype=DNA /annot=exon
10	X14008	Lysozyme gene (EC 3.2.1.17)
10	X17648	GRANULOCYTE-MACROPHAGE COLONY-STIMULATING FACTOR RECEPTOR ALPHA CHAIN PRECURSOR
10	U22376	C-myb gene extracted from Human (c-myb) gene, complete primary cds, and five complete alternatively spliced cds
10	M32304	TIMP2 Tissue inhibitor of metalloproteinase 2
10	X70297	CHRNA7 Cholinergic receptor, nicotinic, alpha polypeptide 7
10	M63838	Interferon-gamma induced protein (IFI 16) gene
10	M33195	Fc-epsilon-receptor gamma-chain mRNA
10	S82470	BB1

TNoM	Gene	Description
11	Z49194	OBF-1 mRNA for octamer binding factor 1
11	L47738	Inducible protein mRNA
11	X07743	PLECKSTRIN
11	M29696	IL7R Interleukin 7 receptor
11	L21954	PERIPHERAL-TYPE BENZODIAZEPINE RECEPTOR
11	M31166	PTX3 Pentaxin-related gene, rapidly induced by IL-1 beta
11	X98411	GB DEF = Myosin-IE
11	X97267	LPAP gene
11	Y07604	Nucleoside-diphosphate kinase
11	J02783	P4HB Procollagen-proline, 2-oxoglutarate 4-dioxygenase (proline 4-hydroxylase), beta polypeptide (protein disulfide isomerase; thyroid hormone binding protein p55)
11	Z29067	Nek3 mRNA for protein kinase
11	L09717	LAMP2 Lysosome-associated membrane protein 2 alternative products
11	U02020	Pre-B cell enhancing factor (PBEF) mRNA
11	X76648	GLRX Glutaredoxin (thioltransferase)
11	L42379	Quiescin (Q6) mRNA, partial cds
11	M13792	ADA Adenosine deaminase
11	U16954	(AF1q) mRNA
11	X06182	KIT V-kit Hardy-Zuckerman 4 feline sarcoma viral oncogene homolog
11	M81695	ITGAX Integrin, alpha X (antigen CD11C (p150), alpha polypeptide)
11	U70063	Acid ceramidase mRNA
11	J04990	CATHEPSIN G PRECURSOR
11	M23178	MACROPHAGE INFLAMMATORY PROTEIN 1-ALPHA PRECURSOR
11	M98399	CD36 CD36 antigen (collagen type I receptor, thrombospondin receptor)
12	M89957	IGB Immunoglobulin-associated beta (B29)
12	M29474	Recombination activating protein (RAG-1) gene
12	HG2788-HT2896	Calcyclin
12	M92357	B94 PROTEIN
12	M19045	LYZ Lysozyme
12	S50223	HKR-T1
12	X66401	LMP2 gene extracted from H.sapiens genes TAP1, TAP2, LMP2, LMP7 and DOB
12	U49020	MEF2A gene (myocyte-specific enhancer factor 2A, C9 form) extracted from Human myocyte-specific enhancer factor 2A (MEF2A) gene, first coding
12	D10495	PRKCD Protein kinase C, delta
12	U60644	HU-K4 mRNA
12	M28130	Interleukin 8 (IL8) gene
12	Y00787	INTERLEUKIN-8 PRECURSOR
12	L06797	PROBABLE G PROTEIN-COUPLED RECEPTOR LCR1 HOMOLOG
12	D49950	Liver mRNA for interferon-gamma inducing factor(IGIF)
12	U41813	HOXA9 Homeo box A9
12	U97105	Dihydropyrimidinase related protein-2
12	X63469	GTF2E2 General transcription factor TFIIE beta subunit, 34 kD
12	X74262	RETINOBLASTOMA BINDING PROTEIN P48
12	L11672	ZNF91 Zinc finger protein 91 (HPF7, HTF10)
12	HG4321-HT4591	Ahnak-Related Sequence
12	U40369	Spermidine/spermine N1-acetyltransferase (SSAT) gene
12	M13452	LMNA Lamin A
12	X58431	HOX 2.2 gene extracted from Human Hox2.2 gene for a homeobox protein
12	AB002559	Hunc18b2
12	M54995	PPBP Connective tissue activation peptide III
12	U00802	Drebrin E
13	M84371	CD19 gene
13	U29175	Transcriptional activator hSNF2b
13	K01911	NPY Neuropeptide Y
13	M28170	CD19 CD19 antigen
13	D26156	Transcriptional activator hSNF2b
13	M63959	LRPAP1 Low density lipoprotein-related protein-associated protein 1 (alpha-2-macroglobulin receptor-associated protein 1
13	M65214	TCF3 Transcription factor 3 (E2A immunoglobulin enhancer binding factors E12/E47)
13	M80254	PEPTIDYL-PROLYL CIS-TRANS ISOMERASE, MITOCHONDRIAL PRECURSOR

TNoM	Gene	Description
13	J03801	LYZ Lysozyme
13	M95678	PLCB2 Phospholipase C, beta 2
13	U57721	L-kynurenine hydrolase mRNA
13	L11669	Tetracycline transporter-like protein mRNA
13	L19437	TALDO Transaldolase
13	L41559	PCBD 6-pyruvoyl-tetrahydropterin synthase/dimerization cofactor of hepatocyte nuclear factor 1 alpha (TCF1)
13	M33680	26-kDa cell surface protein TAPA-1 mRNA
13	U72621	LOT1 mRNA
13	X80230	mRNA (clone C-2k) mRNA for serine/threonine protein kinase
13	X77533	Activin type II receptor
13	U16306	CSPG2 Chondroitin sulfate proteoglycan 2 (versican)
13	U07139	CAB3b mRNA for calcium channel beta3 subunit
13	J04615	SNRPN Small nuclear ribonucleoprotein polypeptide N
13	X15414	ALDR1 Aldehyde reductase 1 (low Km aldose reductase)
13	D38073	MCM3 Minichromosome maintenance deficient (S. cerevisiae) 3
13	M21535	GB DEF = Erg protein (ets-related gene) mRNA
13	X51521	VIL2 Villin 2 (ezrin)
13	M29971	MGMT 6-O-methylguanine-DNA methyltransferase (MGMT)
13	X16706	FOS-RELATED ANTIGEN 2
13	M57731	GRO2 GRO2 oncogene
13	X52192	FES Feline sarcoma (Snyder-Theilen) viral (v-fes)/Fujinami avian sarcoma (PRCII) viral (v-fps) oncogene homolog
13	X91911	Glioma pathogenesis-related protein (GliPR) mRNA
13	U46006	GB DEF = Smooth muscle LIM protein (h-SmLIM) mRNA
13	X79067	ERF-1 mRNA 3' end
13	L41162	COL9A3 Collagen, type IX, alpha 3
13	U29656	NME1 Non-metastatic cells 1, protein (NM23A) expressed in
13	U41635	OS-9 precurosor mRNA
13	U83600	GB DEF = Death domain receptor 3 (DDR3) mRNA, alternatively spliced form 2, partial cds
13	X59711	NFYA Nuclear transcription factor Y, alpha
13	M20203	GB DEF = Neutrophil elastase gene, exon 5
13	D26308	NADPH-flavin reductase
13	J03589	UBIQUITIN-LIKE PROTEIN GDX
13	X55668	PRTN3 Proteinase 3 (serine proteinase, neutrophil, Wegener granulomatosis autoantigen)