Available online at www.sciencedirect.com
SCIENCE DIRECT®
Pattern Recognition Letters 26 (2005) 1444–1453
Pattern Recognition Letters
www.elsevier.com/locate/patrec

Feature selection algorithms to find strong genes

Paulo J.S. Silva [a,*], Ronaldo F. Hashimoto [a], Seungchan Kim [b], Junior Barrera [a], Leônidas O. Brandão [a], Edward Suh [c], Edward R. Dougherty [d]

[a] *Department of Computer Science, Institute of Math.; Statistics—IME, University of São Paulo, Rua do Matao 1010, 05508-090 Sao Paulo, Brazil*
[b] *Cancer Genetics Branch, National Human Genome Research Institute, National Institutes of Health, Bethesda, MD 20892-4470, USA*
[c] *Division of Computational Biology, Center for Information Technology, National Institutes of Health, Bethesda, MD 20892-4470, USA*
[d] *Department of Electrical Engineering, Texas A&M University, College Station, TX 77840, USA*

Received 2 October 2004
Available online 30 December 2004

Abstract

The cDNA microarray technology allows us to estimate the expression of thousands of genes of a given tissue. It is natural then to use such information to classify different cell states, like healthy or diseased, or one particular type of cancer or another. However, usually the number of microarray samples is very small and leads to a classification problem with only tens of samples and thousands of features. Recently, Kim et al. proposed to use a parameterized distribution based on the original sample set as a way to attenuate such difficulty. Genes that contribute to good classifiers in such setting are called *strong*. In this paper, we investigate how to use feature selection techniques to speed up the quest for strong genes. The idea is to use a feature selection algorithm to filter the gene set considered before the original strong feature technique, that is based on a combinatorial search. The filtering helps us to find very good strong gene sets, without resorting to super computers. We have tested several filter options and compared the strong genes obtained with the ones got by the original full combinatorial search.

Keywords: Feature selection; Strong genes; Gene expression analysis; High-dimensional problem; Small sample size

1. Introduction

There are many ways to design a classifier from sample data. The worth of such classifier depends on the suitability of the particular classification rule to the feature-label distribution and the amount of sample data. Since we are in the context of very

* Corresponding author. Tel.: +55 11 3091 5178; fax: +55 11 3091 6134.
E-mail address: pjssilva@ime.usp.br (P.J.S. Silva).

0167-8655/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2004.11.017

small sample size, the latter issue is our main concern in this work. This lack of training data makes it necessary to apply simple classification rules to avoid overfitting. Here we are interested in the particular case in which the data are linear separable, and therefore a linear classifier (perceptron model) is suitable for classification. Although ordinarily such an assumption might seem too strong, it has proven to be the case in many real world applications coming from gene expression analysis of cancer data (for instance see Kim et al., 2002a,b; Morikawa et al., 2003; Luo et al., 2003; Bomprezzi et al., 2003; Simon, 2003).

The error rate of designed classifiers over the population of samples tends to have a large variance in a small-sample setting. Hence, the selection of feature sets is problematic. Given a large set of potential features, it is necessary to find a small subset that provides good classification. Small feature sets are prudent because the complexity of the classifier, and therefore the data requirement, typically grows with increasing numbers of features. Even with small feature sets, if there are thousands of features, then the number of possible feature sets can be astronomical. Hence, even if the classes are moderately separated, for small samples there may be thousands of feature sets whose error estimates are close to zero. It would be wrong to conclude that the true errors of all corresponding classifiers are small. Low estimation of a classifier's population error can result from the peculiarity of the particular sample and/or the large deviation between the true error of the designed classifier and its estimated error computed from the same training data from which it has been designed. In our application of direct interest, cancer classification, cross-validation error estimation has been very popular; however, there are serious questions with regard to its application in the case of very small settings (Braga-Neto et al., 2004; Braga-Neto and Dougherty, 2004a,b).

To lower both the risks of overfitting and choosing a feature set based on a low error estimate, rather than design a classifier directly from a small sample, Kim et al. (2002a) have recently proposed to design a perceptron from a distribution based on the sample and for which it is more difficult to distinguish the labels. This is done in a parameterized manner in which the parameter relates to the difficulty of classification. The resulting features are called "strong features." In the case of perceptrons, a strictly analytic approach is used to find both the classifier and its error. Analytic design and error estimation facilitate efficient computation in the context of a large set of potential features; nonetheless, the computation becomes quickly impossible and the method has been applied using a supercomputer. Here we consider the efficacy of several different feature-selection methods in the context of the strong-feature algorithm.

The immediate application of interest is classification via cDNA microarrays, which provide expression measurements for thousands of genes simultaneously (DeRisi et al., 1997; Duggan et al., 1999; Schena et al., 1995). A key goal for the use of expression data is to perform classification via different expression patterns. A successful classifier provides a list of genes whose product abundance is indicative of important differences in cell state, such as healthy or diseased, or one particular type of cancer or another. Two central goals of molecular analysis of disease are to use such information to directly diagnose the presence or type of disease and to produce therapies based on the disruption or correction of the aberrant function of gene products whose activities are central to the pathology of a disease. Correction would be accomplished either by the use of drugs already known to act on these gene products or by developing new drugs targeting these gene products. Achieving these goals requires designing a classifier that takes a vector of gene expression levels as input and outputs a class label, which predicts the class containing the input vector. Classification can be between different kinds of cancer, different stages of tumor development, or many other such differences.

The inherent class-separating power of expression data has been clearly demonstrated (Ben-Dor et al., 2001; Golub et al., 1999; Hedenfalk et al., 2001; Khan et al., 2001; Kobayashi et al., 2003). Going further, sufficient information must be vested in sets of genes small enough to serve as either convenient diagnostic panels or as candidates for the very expensive and time-consuming analysis required to determine if they could serve

as useful targets for therapy. The problem at this stage is that there is a very large set of gene-expression profiles (features) and typically a small number of microarrays (sample points), making it difficult to find the best features from which to construct a classifier. We require methods to find gene sets that can perform accurate classification in distributional settings whose dispersions are in excess of the sample data. In this direction, the strong-feature methodology has been used successfully to find feature sets in several oncogenomic settings: breast cancer (Kim et al., 2002a), glioma (Kim et al., 2002b), lymphoma (Kobayashi et al., 2003), and leukemia (Morikawa et al., 2003). The purpose of this paper is to examine the performance of a number of feature-selection methods for the strong-feature methodology.

2. Finding strong features: original algorithm

In this section, we review the strong-feature algorithm. We denote random variables by capital italic letters $A, B, \ldots, Z$. A random vector will be denoted by a capital italic boldface letter. For example $\boldsymbol{X} = (X_1, X_2, \ldots, X_d)$. A binary classification involving the random feature vector $\boldsymbol{X}$ is determined by a binary random variable Y taking the values (*class labels*) 0 and 1. A *classifier* or *filter* is a function of $\boldsymbol{X}$ which is an estimator of Y. For a feature vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$, a perceptron is defined by $T(\langle \boldsymbol{w}, \boldsymbol{x} \rangle - \gamma)$, where $\boldsymbol{w} \in \mathbb{R}^d$, $\langle \cdot, \cdot \rangle$ denotes the usual inner product, and T is a threshold function, $T(z) = 0$ if $z \leqslant 0$, and $T(z) = 1$ if $z > 0$. That is, the perceptron splits $\mathbb{R}^d$ into two regions separated by the hyperplane $\langle \boldsymbol{w}, \boldsymbol{x} \rangle = \gamma$. Hence, its design requires estimating the normal to the hyperplane w and its displacement coefficient γ.

In order to lower the risk of choosing a feature set based on a low error estimate, rather than design a classifier (perceptron) directly from a small sample, Kim et al. (2002a) propose in the original strong feature paper to design the perceptron from a distribution based on the sample. This is done by spreading the sample data using an independent random distribution parameterized by its variance σ^2. The bigger σ^2, the more difficult is the classification. Since the spread data are parameterized by the variance σ^2, so is the resulting optimal perceptron, ψ_σ, which is called the *σ-perceptron*. The error, ε_σ, for ψ_σ is called the σ-error and is computed analytically from the defining hyperplane. For $\sigma = 0$, there is no spreading of the sample mass and ε_σ is equal to the resubstitution error estimate for the sample. The strength of the feature vector $\boldsymbol{X}$ relative to the sample and spread distribution is defined by $\zeta_{\boldsymbol{X}}(\sigma) = 1 - \varepsilon_\sigma$.

When designing a classifier from training data, it is necessary to choose values of the spread σ. A simple approach is to set a threshold for the σ-error and push σ as high as possible while keeping the error below the threshold. A systematic approach is to derive a dispersion value for the sample data and use that value to arrive at normalized spread values. For feature vector $\boldsymbol{X} = (X_1, X_2, \ldots, X_d)$, let $\sigma_{k,C}$ be the standard deviation of X_k on the class C, and let $\sigma_{\max}$ be the maximum over all $\sigma_{k,C}$ for both classes. In practice, $\sigma_{k,C}$ is estimated from the training data. Choosing a normalized spread, σ_{nor}, between 0 and 1, a corresponding spread for the sample data is obtained by $\sigma = \sigma_{\text{nor}}\sigma_{\max}$. Using $\sigma_{\max}$ for the normalization maintains a conservative attitude towards estimation of misclassification error. Other approaches are possible. For instance, when the number of points in a class is very small, decent estimation of $\sigma_{k,C}$ is not possible, so the variances might be pooled in some way. The effects of different values of σ_{nor} have been considered in the context of a model problem, and in that context $\sigma_{\text{nor}} = 0.6$ provides a conservative estimate of the true error, with values exceeding 0.6 providing more conservative estimates (Kim et al., 2002a). Another approach to find good values for σ_{nor} is presented in (Braga-Neto and Dougherty, 2004a). There, the authors show how to choose the parameter in order to get nearly unbiased error estimates. Henceforth, we will drop the subscript and in all cases let σ denote the normalized spread.

An exhaustive search to find strong feature sets for increasing values of σ is combinatorial. For large gene sets in the thousands, the full search can only be implemented using supercomputers for $d \leqslant 3$. For larger sets, or to decrease the computational effort, Kim et al. (2002a) utilized a guided random walk implementation. The algo-

rithms discussed in the paper try to reduce the amount of work needed to make it possible to use an ordinary workstation for $d = 3$, and make the method feasible for $d > 3$. We restrict our attention to $d = 3$ for two reasons: first, it allows comparison with the full search; second, given the small samples involved, a maximum of 3-gene classification is prudent, and this is what has been done in the applications.

3. Short description of the feature selection methods

In order to save computational effort in the exhaustive search to find strong feature sets, we propose to use a feature selection algorithm to decrease the number of genes to a manageable amount. After this first pre-selection phase, full search should be attempted on the pre-selected genes in order to recover the best strong gene sets. Following this strategy, we hope to be able to find many of the best sets, while decreasing considerably the required computational resources.

We present below 5 feature selection algorithms that may be used in the pre-selection phase. Two algorithms come from the bioinformatics literature: guided random walk and ranked gene list. The guided random walk was proposed by Kim et al. (2002a) in the original strong gene paper to save computational resources. Ranked gene lists have been used by Hedenfalk et al. (2001) to find differentially expressed genes. The next three methods come from the feature selection literature. The first, principal component analysis, is one of the most popular dimensionality reduction techniques. However, as it builds artificial features from the original ones, it may not be well suited for our application. Next, we present the sequential floating search method (SFFS) that is very popular in the feature selection community (Jain and Zongker, 1997; Pudil et al., 1994). Finally, there is the linear support vector machine (SVM). It possesses a character akin to the strong feature definition that we will explore.

3.1. Guided random walk

This method was proposed in the original strong features paper (Kim et al., 2002a, Section 5). It uses probabilistic information build from a previous full search considering a very small number of genes, typically 2. The main idea is based on the assumption that if a feature is part of a small good feature set, it will be more likely that the same feature enters in a larger good set. Hence, the algorithm gives a higher probability to accept those features. Moreover, the method tries to explore different regions of the search space to find many good solutions.

3.2. Ranked gene list

In this approach (Hedenfalk et al., 2001), each gene is weighted based on its discriminative ability for two distinct classes. This discriminative weight is evaluated according to a gene's impact on minimizing its class cluster volume and maximizing center-to-center inter-class distance. More formally, let $\{\boldsymbol{x}^{0,1}, \boldsymbol{x}^{0,2}, \ldots, \boldsymbol{x}^{0,n_0}\}$ and $\{\boldsymbol{x}^{1,1}, \boldsymbol{x}^{1,2}, \ldots, \boldsymbol{x}^{1,n_1}\}$ be the set of samples labelled 0 and 1 respectively. The weight for each feature is evaluated by

$$dc_i := \left| \frac{\sum_{j=1}^{n_0} x_i^{0,j}}{n_0} - \frac{\sum_{j=1}^{n_1} x_i^{1,j}}{n_1} \right| \tag{1}$$

$$md_i^0 := \sum_{j \neq l} |x_i^{0,j} - x_i^{0,l}|$$

$$md_i^1 := \sum_{j \neq l} |x_i^{1,j} - x_i^{1,l}| \tag{2}$$

$$md_i := \frac{md_i^0 + md_i^1}{\binom{n_0}{2} + \binom{n_1}{2}} \tag{3}$$

$$w_i := \frac{dc_i}{md_i + \alpha} \tag{4}$$

where for a given gene i, dc_i is the distance between the mean expressions of both classes; md_i^0 and md_i^1 are accumulated distances between the expressions of all pairs in the same class; md_i is the mean distance between the expressions of pairs in the same class; and w_i is the final weight associated to the gene. The constant $\alpha > 0$ should be small, and is only used to avoid division by zero. It is easy to

see that w_i increases if the prototypes for both classes are far from each other. On the other hand, it decreases whenever the classes are not closely packed.

3.3. Principal component analysis

Although the terms "feature selection" and "feature extraction" are often used interchangeably in the literature, it is important to make a distinction between them. The term "feature selection" refers to algorithms that select the best subset of the input feature set. Methods that create new features based on transformations or combinations of the original features are called "feature extraction."

As in Section 2, let d denote the number of features present in n sample vectors. One of the better known linear feature extractors is the principal component analysis (PCA), or the Karnhunen–Loève expansion. It computes the m largest eigenvectors of the $d \times d$ covariance matrix of the samples, and use them as a base for a m-dimensional space that captures most of the variation present in the original data. Hence, the PCA generates new features (linear combinations of given features) that are the most expressive ones (eigenvectors with the largest eigenvalues). Although these new features may possess discriminative ability, they may not have a clear physical meaning. This is especially important in gene-expression-based classification.

To go back to the original features, one may try to identify the coordinates of the principal components that have the most expressive values, that is, the largest absolute values. Given a principal component, this may be done as follows:

(1) Divide the interval between the minimum and the maximum absolute values of the coordinates of the given principal component in 100 parts. More formally, find $\tau_1, \ldots, \tau_{101}$ equally spaced such that τ_1 and τ_{101} are the minimum and the maximum absolute values of the coordinates respectively.
(2) For each τ_i compute the number of coordinates whose absolute value is larger than τ_i. This number will be called μ_i.
(3) Find the largest τ_i such that $\mu_i - \mu_{i+1}$ is strictly greater than a given lower bound. This lower bound is certainly subjective and in our experiments we have defined it to be 2.
(4) Select all the coordinates whose absolute value is larger than τ_i selected above.

Hence, for each principal component, we have a feature subset. Then, the feature set for the full-search step is the union of all those feature subsets.

3.4. Sequential forward floating method (SFFS)

This method was introduced by Pudil et al. (1994) and it is very popular among the feature selection community (Jain and Zongker, 1997). It starts with an empty feature set and then it tries to improve the set by adding a and removing features at each step based on the new set quality. The method is a development of previous efforts like the "Plus l—Take away r" (Somol et al., 1999) where the number of features that will be added and removed are dynamically computed at each step.

The quality measure required by the SFFS was defined to be the error given by the strong-feature model for each gene set. Note that this error is easy to compute since it is based on the solution of a small linear equation, as described in (Kim et al., 2002a, Section 4). Note that this implies that the selected genes depend on the parameter σ.

The SFFS is controlled by some parameters that have to be supplied beforehand. First, we need to choose the target size for the group of genes that should be pre-selected. A naive approach would be to ask the SFFS to do all the pre-selection in one step, i.e. ask it to search for a group of approximately 100 genes. This is not desirable, since the method takes too long to complete with such large group sizes. On the other hand, it is interesting to explore groups with many genes, as we are particularly interested in genes that combine with other genes to generate good classifiers. After some computational experiments, we have settled on running the SFFS to find groups of 10 genes that work well together. We run then the SFFS many times to gather the desired number of genes in the pre-selection phase. Another important choice is the δ parameter that

controls when the floating search may stop. We have used $\delta = 3$. These parameters showed the best compromise between the quality of the pre-selected genes, considering the best triples that could be found among them, and computational time required by the pre-selection step.

3.5. Linear support vector machine (SVM)

If we consider the definition of feature strength given in Section 2, we can observe that it is akin to the idea of support vector machines (Vapnik, 1995). Given a classification hyperplane, the error associated to each sample that is correctly classified is negatively correlated to its distance to the hyperplane. Therefore, if we can find a hyperplane that correctly classifies all the data and is far from both sample sets, it should present high strength. Of course, it may well be that it is possible to find sufficiently strong classifiers when the sample data are not linearly separable, and not find them when the data are linearly separable. *The point here is that strong classification tends to be more achievable for linearly separable data*, and, for the gene-expression data we are considering, there is typically a large number of gene sets for which the data arising from the different disease types are linearly separable.

To effectively use the SVM, we need to ensure that it will select the best-separating features (genes). The idea of using support vector machines for feature selection was introduced by Bradley et al. (1998). In order to induce feature selection they have proposed to change the space norm: instead of using the common Euclidean norm, they suggest to use the infinity norm.

Let $\boldsymbol{X}^0 \in \mathbb{R}^{n_0 \times d}$ be a real matrix that has as each row the sample vectors $\boldsymbol{x}^{0,1}, \boldsymbol{x}^{0,2}, \ldots, \boldsymbol{x}^{0,n_0}$ transposed. Analogously, $\boldsymbol{X}^1 \in \mathbb{R}^{n_1 \times d}$ is a matrix composed by the samples labelled 1. As indicated in (Bradley et al., 1998), to find the best separating hyperplane with respect to the infinity norm we must solve the following optimization problem:

$$\begin{aligned} \min \quad & \sum_{k=1}^{d} |w_k| \\ \text{s.t.} \quad & \boldsymbol{X}^0 \boldsymbol{w} \geqslant \gamma \boldsymbol{e} + \boldsymbol{e} \\ & \boldsymbol{X}^1 \boldsymbol{w} \leqslant \gamma \boldsymbol{e} - \boldsymbol{e} \end{aligned} \tag{5}$$

where $\boldsymbol{e}$ denotes a vector of 1s of appropriate dimension.

Moreover, to make problem 5 more effective to find strong genes we must take into account the way the spread increases with the standard deviation of the selected features. As described in Section 2, we should prefer features with small inner-class standard deviation. We can use the standard deviation as a penalty parameter in the objective function to achieve this effect. Letting $\sigma_{k,0}$ and $\sigma_{k,1}$ be the standard deviations of the samples labelled 0 and 1, the problem we actually solve is

$$\begin{aligned} \min \quad & \sum_{k=1}^{d} \max\{\sigma_{k,0}, \sigma_{k,1}\} |w_k| \\ \text{s.t.} \quad & \boldsymbol{X}^0 \boldsymbol{w} \geqslant \gamma \boldsymbol{e} + \boldsymbol{e} \\ & \boldsymbol{X}^1 \boldsymbol{w} \leqslant \gamma \boldsymbol{e} - \boldsymbol{e} \end{aligned} \tag{6}$$

This problem may be effectively solved if we introduce the change of variables $w_k^+ - w_-^k := w_k$, where $w_k^+ \geqslant 0$ and $w_k^- \geqslant 0$ represent the positive and the negative parts of w_k. Then, 6 becomes

$$\begin{aligned} \min \quad & \sum_{k=1}^{d} \max\{\sigma_{k,0}, \sigma_{k,1}\} (w_k^+ - w_k^-) \\ \text{s.t.} \quad & \boldsymbol{X}^0 \boldsymbol{w}^+ - \boldsymbol{X}^0 \boldsymbol{w}^- \geqslant \gamma \boldsymbol{e} + \boldsymbol{e} \\ & \boldsymbol{X}^1 \boldsymbol{w}^+ - \boldsymbol{X}^1 \boldsymbol{w}^- \leqslant \gamma \boldsymbol{e} - \boldsymbol{e} \\ & \boldsymbol{w}^+, \boldsymbol{w}^- \geqslant 0 \end{aligned} \tag{7}$$

This is a linear programming problem with $2d + 1$ variables and only n constraints. Note that the number of variables is quite large, since it is equal to twice the number of genes in the array. On the other hand, the number of constraints is small and coincides with the number of array samples.

The small number of constraints have two interesting consequences:

(1) A typical method to solve 7 should be very efficient as the linear algebra will deal with $n \times n$ matrices.
(2) The maximum number of genes that can be selected by a simplex method is n, since this is the size of a basic solution. This number is usually very small when compared to the total

number of genes in the array. In our tests, the number of selected genes is even smaller, usually close to $n/2$.

4. Computational experiments

We have performed a number of computational experiments based on the different proposed strategies to find strong gene sets. In all experiments, the full gene list has been reduced to approximately 100 genes using the methods described in Section 3, after which a full search is employed to find the best feature sets using the pre-selected genes. The results are easily split in two: the guided random walk and PCA do not succeed on finding good gene sets; on the other hand, the other methods—ranked gene list, SFFS, and linear SVM—perform very well even if compared to a full search of all subsets. We consider two cancer–related gene-expression data sets based on cDNA microarrays.

The BRCA data set (Hedenfalk et al., 2001) has also been used in the original strong-gene study (Kim et al., 2002a). It involves expression profiles of breast tumors from patients carrying mutations in the predisposing genes, BRCA1 (7 samples) or BRCA2 (8 samples), or from patients not expected to carry a hereditary predisposing mutation (7 samples). Starting with 3226 genes we apply the strong-feature algorithm with feature selection to subsets of size 3 to find perceptrons for four classification problems: (1) BRCA1 tumors versus the collection of both BRCA2 and sporadic tumors; (2) BRCA2 tumors versus the collection of both BRCA1 and sporadic tumors; (3) BRCA1 tumors versus the BRCA2 tumors, ignoring the sporadic tumors; and (4) BRCA1 tumors and a possibly misclassified sporadic tumor versus the collection of both BRCA2 and other sporadic tumors.

The second data set has been used to classify childhood small, round blue cell tumors (SRBCTs) into four cancer types: neuroblastoma (NB, 12 samples), rhabdomyosarcoma (RMS, 20 samples), Burkitt's lymphoma (BL, 8 samples), and the Ewing tumor family (EWS, 23 samples) (Khan et al., 2001). There are 6567 genes on each microarray, and omitting genes that do not satisfy a threshold level of expression reduces the number to 2308. We use the 63 microarrays used for training in the original study. We consider four classification problems: (1) BL versus others; (2) NB versus others; (3) EWS versus others; and (4) RMS versus others.

The experiments have been run for different values of σ: 0.4, 0.5, 0.6, and 0.7. The results are presented using graphics showing the error of the best triples (triples with smallest σ-error). In the graphics, the vertical axis represent the error and the horizontal axis the position of the best triples. Therefore, if for a given method the error of the triple in position 5 is 0.005, then 0.005 is the σ-error of the fifth best triple. All results, together with related information, such as the pre-selected genes and the best performing groups, are presented on the associated website at http://www.vision.ime.usp.br/strong_features. For the sake of completeness we present typical graphs for the results below.

As it can be seen in Fig. 1, pre-selection of the genes using ranked gene list, SFFS, and linear SVM is very successful on finding the best possible triples. Actually, the result is surprisingly good if we consider that the the amount of triples investigated after the pre-selection is four orders of magnitude smaller than the number of triples visited by full search. We believe that for highly separable data it is very convenient to try first one of these pre-selection strategies.

On the other hand, as shown by Fig. 2, both PCA and guided random walk fail to recover the best triples. This result is somewhat expected for the PCA, as it is a feature extraction method, instead of a feature selection algorithm. However the result for the guided random walk is unexpected since it was the method proposed by Kim et al. (2002a) in the original strong gene paper to save computational effort. We believe that both methods should be avoided in the strong gene search.

Another way to validate the quality of the genes that are pre-selected by the above methods is to compare them with the genes that are present more often in the best triples found by full search. This is done in Table 1 below. For each test case we have selected the 25 genes that appear more often in the

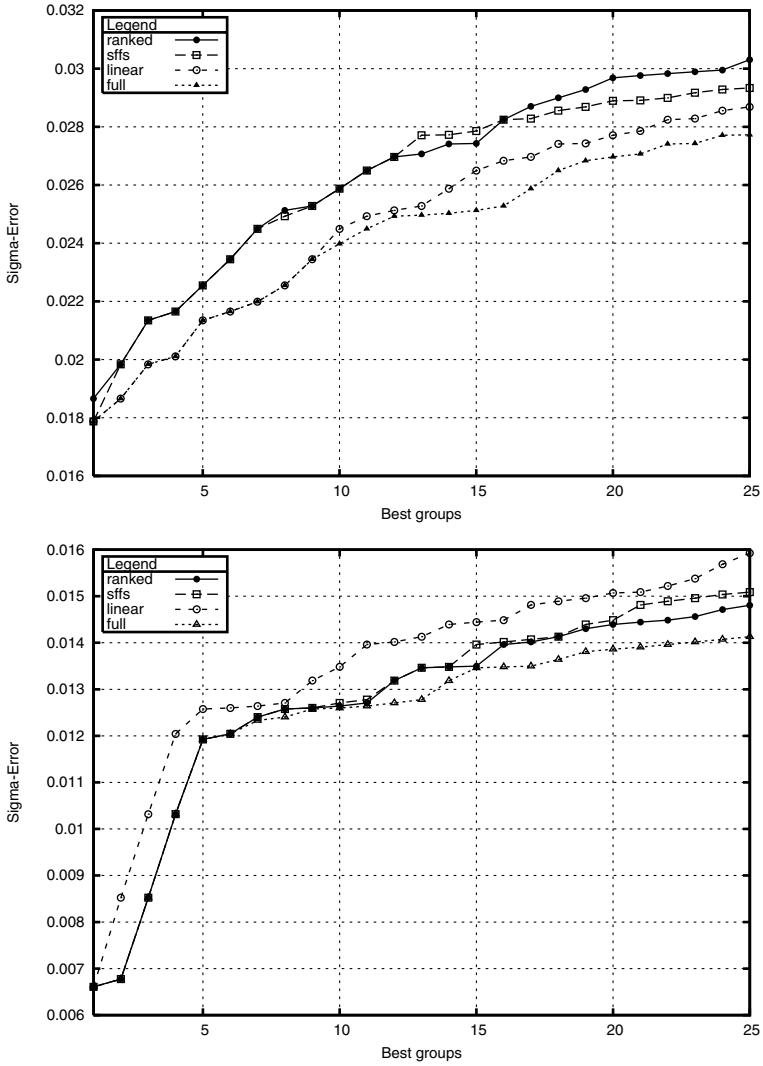

Fig. 1. Computational test comparing the performance of ranked list, SFFS, linear SVM, and full search (no pre-selection). The first graph presents the result for the separation of the BRCA2 class from the others in the BRCA data. The second graph represents the separation of NB class from the others in the SRBCT data. In both cases $\sigma = 0.6$.

100 best triples. At each entry we present the number of those 25 genes that were pre-selected by each pre-selection method.

Once again, we can see that the SFFS, linear SVM and ranked gene list are very successful on finding the best genes. On the other hand, PCA performance is very poor.

Finally, Table 2 below compares the computational times required by the best pre-selection methods and full search. Full search was performed using a Pentium III 800 MHz cluster node. All the other tests were done in a Athlon 1.2 GHz desktop. Both processors are roughly comparable, with the Athlon based computer being a little faster, but certainly not more than 1.5 faster than the Pentium computer. For each experiment we present three time values. First the pre-selection time. Then, the time required to explore with full search the

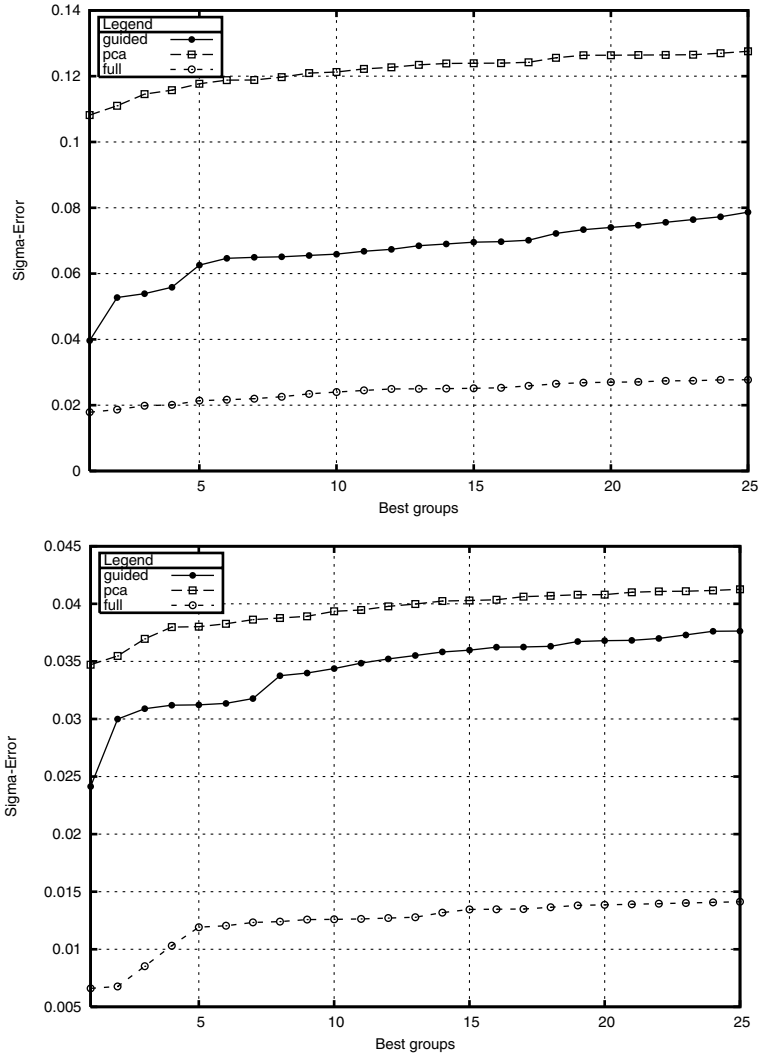

Fig. 2. Computational test comparing the performance of ranked list, SFFS, linear SVM, and full search (no pre-selection). The data are the same used to generate Fig. 1.

Table 1
Most common genes retrieved by the pre-selection

Test case	SFFS	Linear	Ranked	PCA
BRCA2 from others	20	22	20	0
NB from others	12	13	16	2

pre-selected genes. Finally, we add up the times of these two phases. Times are presented in minutes.

As expected, the computational saving for the pre-selection algorithms are huge. They are

Table 2
Computational time for two test samples

Test case	Full	SFFS	Linear	Ranked
BRCA2 from others				
Pre-selection		2.00	0.99	0.19
Comb. search	8654.38	0.10	0.07	0.07
Total	8654.38	2.10	1.16	0.26
NB from others				
Pre-selection		5.45	2.36	1.30
Comb. search	2009.48	0.33	0.53	0.52
Total	2009.48	5.78	2.89	1.82

roughly three orders of magnitude faster. This is a direct result of pre-selection, where the search space is drastically decreased. This fact, combined with the quality of the results detailed above, indicates that the pre-selection strategy may be a very effective way to address strong gene finding.

As expected, the ranked gene list is the fastest. However this method does not explore combinations of genes and may face more difficulty if the classes are less separable. The other methods, linear SVM and SFFS, are both very fast, exploring many gene combinations in minutes.

References

Ben-Dor, A., Bruhn, L., Friedman, N., Nachman, M.S., Yakhini, Z., 2001. Tissue classification with gene expression profiles. Journal of Computational Biology 7 (3–4), 559–583.

Bomprezzi, R., Ringner, M., Kim, S., Bittner, M., Khan, J., Chen, Y., Elkahloun, A., Yu, A., Bielekova, B., Meltzer, P., Marlin, R., McFarland, H., Trent, J., 2003. Gene expression in multiple sclerosis patients and healthy controls: identifying pathways relevant to disease. Human Molecular Genetics 12 (17), 2191–2199.

Bradley, P., Mangasarian, O., Street, W., 1998. Feature selection via mathematical programming. INFORMS Journal on Computing 10, 209–217.

Braga-Neto, U., Dougherty, E., 2004a. Bolstered error estimation. Pattern Recognition 37, 1267–1281.

Braga-Neto, U., Dougherty, E., 2004b. Is cross-validation valid for small-sample microarray classification? Bioinformatics, in press.

Braga-Neto, U., Hashimoto, R., Dougherty, E., Nguyen, D., Carrol, R., 2004. Is cross-validation better than resubstitution for ranking genes? Bioinformatics 20 (2), 253–258.

DeRisi, J., Iyer, V., Brown, P., 1997. Exploring the metabolic and genetic control of gene expression on a genomic scale. Science 278, 680–686.

Duggan, D., Bittner, M., Chen, Y., Meltzer, P., Trent, J., 1999. Expression profiling using cdna microarray. Nature Genetics 21, 10–14.

Golub, T., Slonim, D., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., 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.

Hedenfalk, I., Duggan, D., Chen, Y., Radmacher, M., Bittner, M., Simon, R., Meltzer, P., Gusterson, B., Esteller, M., Raffeld, M., Yakhini, Y., Ben-Dor, A., Dougherty, E., Kononen, J., Bubendorf, L., Fehrle, W., Pittaluga, S., Gruvberger, S., Loman, N., Johannsson, O., Olsson, H., Wilofond, B., Sauter, G., Kallioniemi, O., Borg, A., Trent, J., 2001. Gene expression profiles in hereditary breast cancer. New England Journal of Medicine 344 (8), 539–548.

Jain, A., Zongker, D., 1997. Feature-selection: evaluation, application and small sample performance. IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (2), 152–157.

Khan, J., Wei, J., Ringnér, M., Sall, L., Ladanyi, M., Westermann, F., Schwab, M., Antonescu, C., Peterson, C., Meltzer, P., 2001. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. Nature Medicine 7 (6), 673–679.

Kim, S., Dougherty, E., Barrera, J., Chen, Y., Bittner, M., Trent, J., 2002a. Strong feature sets from small samples. Journal of Computational Biology 9 (1), 127–146.

Kim, S., Dougherty, E., Shmulevich, I., Hess, K., Trent, S.H., Fuller, G., Zhang, W., 2002b. Identification of combination gene sets for glioma classification. Molecular Cancer Therapeutics 1 (13), 1229–1236.

Kobayashi, T., Yamaguchi, M., Kim, S., Morikawa, J., Ueno, S., Suh, E., Dougherty, E.R., Shmulevich, I., Shiku, H., Zhang, W., 2003. Gene expression profiling identifies strong feature genes that classify *de novo* cd5^{+} and cd5 diffuse large b-cell lymphoma and mantle cell lymphoma. Cancer Research 63 (60–66).

Luo, J., Isaacs, W., Trent, J., Duggan, D., 2003. Looking beyond morphology: Cancer gene expression profiling using dna microarrays. Cancer Investigation 21 (6), 937–949.

Morikawa, J., Li, H., Kim, S., Nishii, K., Ueno, S., Suh, E., Dougherty, E., Shmulevich, I., Shiku, H., Zhang, W., Kobayashi, T., 2003. Identification of signature genes by microarray for acute myeloid leukemia without maturation (fam-m1) and aml with t(15;17)(q22;q12)(pml/rar°). International Journal of Oncology 23 (3), 617–625.

Pudil, P., Novovivová, J., Kittler, J., 1994. Floating search methods in feature selection. Pattern Recognition Letters 15, 1119–1125.

Schena, M., Shalon, D., Davis, R., Brown, P., 1995. Quantitative monitoring of gene expression patterns with a complementary dna microarray. Science 270, 467–470.

Simon, R., 2003. Diagnostic and prognostic prediction using gene expression profiles in high-dimensional microarray data. British Journal of Cancer 89 (9), 1599–1604.

Somol, P., Pudil, P., Nonovicová, J., Paclík, J., 1999. Adaptative floating search methods in feature selection. Pattern Recognition Letters 20, 1157–1163.

Vapnik, V., 1995. The Nature of Statistical Learning. Springer, New York.