Recursive Sample Classification and Gene Selection based on SVM: Method and Software Description

Xuegong Zhang\textsuperscript{1,2,*} and Wing H. Wong\textsuperscript{1,3}

\textsuperscript{1} Department of Biostatistics, Harvard School of Public Health, Boston, MA 02115, USA
\textsuperscript{2} Institute of Bioinformatics / Department of Automation, Tsinghua University, Beijing 100084, China
\textsuperscript{3} Department of Statistics, Harvard University, Cambridge, MA 02138, USA

* email: xzhang@hsph.harvard.edu

Basic Consideration

The gene expression data of a sample is a vector containing the expression levels of many genes in the sample measured simultaneously by DNA microarray. From the viewpoint of pattern recognition, the task of cancer classification based on gene expression data is a pattern classification problem, and the feature vector for the classification is the gene expression vector. However, this problem is an extremely difficult one for many methods, since the feature dimension is usually very high (usually several thousands), and the training samples (samples whose correct classification is known and thus plays the role as supervisors) are usually very scarce (say, around 100 known samples or less).

If directly working in this high dimensional space with limited samples, most conventional pattern recognition algorithms may not work well. Some algorithm (algorithms that involve matrix inversion operation) may not be able to arrive at a solution when the number of samples is less than the dimensionality. For others that can achieve a solution, it may not be able to work well on samples other than that used for training. This is called the generalization problem in pattern recognition and machine learning.

Thus, most previous methods for cancer classification based on gene expression data starts with a feature selection procedure. For example, Golub et al defined a metric for evaluating the correlation of a gene with a classification scheme, thus determining whether the gene is relevant or not (Golub et al, 1999; Slonim et al, 2000). Obviously, this kind of strategy does not take possible correlation and co-action among the genes into consideration. Unless it can be proven that the genes are statistically independent with each other (or orthogonal to each other), the result is far from optimal.

We believe that due to the complex (and unknown) relationship among the genes, it should be better to start with the analysis of the full data set. For this purpose, we should be able to design a good classifier with all the candidate genes. Then the most relevant genes for this classification can be discovered by evaluating the subset of genes that contributes mostly in this classifier. We choose SVM or Support Vector Machine as our classifier, due to its supposed good performance on extremely scarce samples in high-dimensional space.

Support Vector Machine (SVM)

The key idea of SVM is generalization: A classifier need not only to work well on the training samples, but also to...
work equally well on previously unseen samples. Although this is realized long before the appearance of SVM, it is SVM that gives this idea a good implementation. The standard theory and algorithm of SVM can be found in many literatures, such as (Cortes and Vapnik, 1995; 1999a; 1999b; Joachims, 1999), and a popular set of SVM codes by Collobert and Bengio (2001) is available at www.idiap.ch/learning/SVMTorch.html. For the purpose of our description, here we only brief the very basic idea of SVM.

A linear SVM is a separation hyperplane with its separation margin maximized and the number of mis-classified samples minimized. The margin is defined as the distance between the hyperplane and the samples of the two classes that are closest to the hyperplane (among those being correctly classified). By minimizing the number of training errors, SVM seeks good performance on the training data; by maximizing the margin, the generalization ability for future data is optimized. The basic problem of SVM can be written as

$$\min \quad \mathbf{w}, \quad \ell \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \left(\sum_{i=1}^{n} \xi_i\right)$$

subject to

$$y_i[(\mathbf{w} \cdot \mathbf{x}_i) + b] - 1 + \xi_i \geq 0, \quad i = 1, \ldots, n,$$

where $$(\mathbf{x}_i, y_i), \quad i = 1, \ldots, n, \quad \mathbf{x}_i \in \mathbb{R}^d, y_i \in \{+1,-1\}$$ are the training samples, and $C$ is a constant controlling the trade-off between maximizing the margin and minimizing the errors. The decision function is

$$f(\mathbf{x}) = \text{sgn}\{(\mathbf{w} \cdot \mathbf{x}) + b\} = \text{sgn}\left\{\sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + b^*\right\}$$

This optimization problem can be solved by the following dual problem:

$$\max \quad Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

subject to

$$\sum_{i=1}^{n} y_i \alpha_i = 0$$

and $$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n.$$

In the final SVM decision function, only a small part of the coefficients $\alpha_i$ are non-zero. The corresponding training samples are called support vectors, since these samples (and only these samples) support the classification boundary.

Using the idea of kernels, linear SVM can be easily extended to its nonlinear version, by replacing the inner product term $(\mathbf{x}_i \cdot \mathbf{x}_j)$ in (2) and (3) with a proper kernel $K(\mathbf{x}_i, \mathbf{x}_j)$. However, due to the special characteristic of the classification problem of microarray data, that is extremely small samples in extremely high dimensional space, information is far from sufficient for reliably estimating any nonlinear relation. Thus we choose linear SVM in our study, taking it as a reasonable approximation to the truth even if it is of a nonlinear nature. This agrees with the basic principle of the statistical learning theory (Vapnik, 1999a): There are only few samples in a very high dimensional feature space, so the complexity of the classification machine should better be simple to guarantee good generalization ability.

**Ranking the Genes according to Their Contribution**

For linear SVM, the final decision function $f(\mathbf{x})$ is a linear one, which is the weighted sum of all the features (the expression levels of all the genes), plus a constant term as a threshold. If $f(\mathbf{x}) > 0$, then the sample is class 1, otherwise class 2, and the larger the absolute value of $f(\mathbf{x})$ is, the more distinct the sample is from the other class. So the simplest way to check the contribution of a single gene to the classification is to calculate whether the term corresponding to this gene in the decision function tends to enlarge the difference between samples of the two classes or not. According to this criterion, the genes can be sorted in the order of their relative contribution in the
classification function. In our experiments, the mean values of samples in the same class were used when calculating this contribution. (See Fig. 1) The intuitive idea behind this criterion is that we take the two class means (the two pseudo-cases whose expression values are the average of the expression values of all the cases in the same class) as two typical models representing the two classes, and assess the contribution of each gene according to its contribution in separating the two cases as far away as possible.

\[
\sum_{x_i \in \text{class 1}} x_i
\sum_{x_i \in \text{class 2}} x_i
\]

The difference of the two class means in the decision function is:

\[
S = \sum_{i=1}^{n} w_i m_i^+ - \sum_{i=1}^{n} w_i m_i^- = \sum_{i=1}^{n} w_i (m_i^+ - m_i^-).
\]

According to the idea of large-margin in statistical learning theory, a larger \( S \) corresponds better generalization ability. Therefore, if we want to select a subset of genes from all the \( n \) genes, the proper way is to keep those genes that give the largest positive contribution in \( S \). So we define the criterion for ranking the genes as

\[
r_i = w_i (m_i^+ - m_i^-).
\]

**Recursive Classification and Gene Selection**

The selection of an “optimal” subset of features from a feature set is a combinatorial problem, which cannot be solved when the dimension is high without the involvement of certain assumptions or compromise, which results in only suboptimal solutions. Here we use a recursive procedure to approach the problem. The idea is, starting from all the available genes, build an optimal SVM model and rank the genes according to their contribution in this model (on this data set). Select a subset of genes that give the largest contribution and build a new SVM model with these selected genes. Re-rank the selected genes according to their contribution in the new model, and then repeat the selection. This classification-ranking-selection-classification procedure can be done recursively, resulting in a smaller subset of genes in each loop. With this series of selected genes, the performance of SVM can be assessed, and a relatively “optimal” number of genes can be finally decided.

For the current study on the lymph node status prediction of breast cancer, we arbitrarily choose the number of selected genes in the recursive procedure to be 1000, 500, 200, 100, 50, 30 and 20, since our major purpose at this stage is to find evidences that indicate the nodal status is predictable from the expression profile of primary breast cancer. For studies which aim at discovering the subset of genes that dominate the classification problem, finer steps might be a better choice. However, we should keep in mind that the gene selection task in microarray data classification is a really sophisticated one if the aim is to discover all the genes relevant to the classification. A systematic study on this topic is undergoing in our group and the result will be available soon.

**Cross-Validation and Permutation Experiments**

When the sample size is small so that we cannot afford to use an independent test set, cross validation is the usual choice for assessing the performance of the classifier. Leave-one-out cross validation is a typical choice. It should be emphasized that when sample size is small, the
gene selection depends heavily on the specific samples used for the selection, no matter what method is used. The gene selection procedure is a part of the whole classification system. Thus, for the cross validation, the sample to be left out as test sample should be removed from the data set at the very beginning, before any gene selection procedure.

The leave-one-out cross validation procedure that we use is:

1. Leave one case out from the data set;
2. Using the remaining cases in the data set to:
   2.1.1 Build a SVM classification model
   2.1.2 Ranking the genes according to this model
   2.1.3 Select a subset of genes for the next step
   2.1.4 Goto 2.1 with the selected subset of genes
3. Test on the left-out case the series of SVM models obtained in step2 at various gene-selection level;
4. Loop to step1, leave another case out;
5. Count the errors among all the leave-out cases, separately at different gene-selection levels, summarize the error rate at each leave.

To get the statistical significance of the reached classification accuracy, we adopted random permutation experiment to study the probability of getting such error rates by chance on randomly assigned classes on the data set, with the same SVM scheme. The permutation is done by randomly shuffling the class labels in the whole data set (while keeping the proportion of the two classes consistent with the true class-label situation). Then the same leave-one-out cross validation procedure is done on every permuted data set. Typically we run 1000 permutation experiments with different random seeds each time, and then estimate the distribution of the error rate on the permuted data, and calculate the permutation p-value for the error to be equal to or less than the error obtained with the true data set. This is a rather strict way to assess the significance of the classification performance.

The cross validation and permutation procedures are illustrated in the diagram of Fig.2.

The Software Package

We developed a software package for recursive classification and gene selection based on SVM for the general UNIX platform. The core algorithm of SVM we use is SVMTorch, by Collobert, R. and Bengio, S. (2001). The Linux version of the executable code is available at http://www.biostat.harvard.edu/complab/. Please refer to the readme file that comes with the package for usage instructions.

Acknowledgements

This work is supported by the NSFC grant 69885004 of China and NIH grant IR01HG02341.

References

on Computational Molecular Biology (RECOMB), pp. 263-272, Tokyo: Universal Academy Press


Fig.2  The diagram of cross-validation and permutation experiments