Classification

Classification in DNA microarray experiments

Sandrine Dudoit, Jane Fridlyand, and Robert Gentleman
Bioconductor short course
Summer 2002

© Copyright 2002 all rights reserved

Outline

- Introduction to classification in microarray experiments.
- Classification as statistical decision theory.
- Overview of classifiers
  - linear and quadratic discriminant analysis;
  - logistic discrimination;
  - nearest neighbor classifiers;
  - classification trees;
  - support vector machines.

Classification

Classification is a prediction or learning problem in which the variable to be predicted assumes one of K unordered values,
\{c_1, c_2, \ldots, c_K\}, arbitrarily relabeled as \{1, 2, \ldots, K\} or sometimes
\{0, 1, \ldots, K - 1\}.

The \(K\) values correspond to \(K\) predefined classes, e.g., tumor
class, bacteria type.
Classification

Associated with each object are

- a response or dependent variable (class label) \( Y \in \{1, 2, \ldots, K\} \), and
- a set of \( G \) measurements which form the feature vector or vector of predictor variables \( \mathbf{X} = (X_1, \ldots, X_G) \).

The feature vector \( \mathbf{X} \) belongs to a feature space \( \mathcal{X} \) (e.g., the real numbers \( \mathbb{R}^G \)).

The task is to classify an object into one of the \( K \) classes on the basis of an observed measurement \( \mathbf{X} = \mathbf{x} \), i.e., predict \( Y \) from \( \mathbf{X} \).

Unsupervised vs. supervised learning

Unsupervised learning. The classes are unknown a priori and need to be “discovered” from the data, a.k.a. cluster analysis; class discovery; unsupervised pattern recognition.

Supervised learning. The classes are predefined and the task is to understand the basis for the classification from a set of labeled objects (training or learning set). This information is then used to classify future observations, a.k.a. discriminant analysis; class prediction; supervised pattern recognition.

Classification in microarray experiments

- Classification is an important question in microarray experiments, for purposes of classifying biological samples and predicting clinical or other outcomes using gene expression data
  - tumor class: ALL vs. AML, classic vs. desmoplastic medulloblastoma;
  - response to treatment, survival;
  - type of bacterial pathogen, etc.
- Although classification is by no means a new subject in the statistical literature, the large and complex multivariate datasets generated by microarray experiments raise new methodological and computational challenges.

Tumor classification using gene expression data

A reliable and precise classification of tumors is essential for successful diagnosis and treatment of cancer.

Current methods for classifying human malignancies rely on a variety of morphological, clinical, and molecular variables.

In spite of recent progress, there are still uncertainties in diagnosis.

Also, it is likely that the existing classes are heterogeneous and comprise diseases which are molecularly distinct and follow different clinical courses.
Tumor classification using gene expression data

DNA microarrays may be used to characterize the molecular variations among tumors by monitoring gene expression profiles on a genomic scale.

This may lead to a finer and more reliable classification of tumors, and to the identification of marker genes that distinguish among these classes.

Eventual clinical implications include an improved ability to understand and predict cancer survival.

Gene expression data

Features correspond to the expression measures for different genes; classes correspond to different biological outcomes (e.g. ALL vs. AML, survivor vs. non-survivor) and are labeled by \( \{1, 2, \ldots, K\} \).

Gene expression data on \( G \) genes (features) for \( n \) mRNA samples (observations)

\[
x_i = (x_{i1}, x_{i2}, \ldots, x_{iG})
\]

Gene expression profile / feature vector for sample \( i \)

\[
y_i = \text{tumor class / response for sample } i,
\]

\( i = 1, \ldots, n. \)

May have covariates such as age, sex.

Gene expression data on \( G \) genes (features) for \( n \) mRNA samples (observations)

\[
X_{G \times n} = \begin{bmatrix}
    x_{11} & x_{12} & \cdots & x_{1n} \\
    x_{21} & x_{22} & \cdots & x_{2n} \\
    \vdots & \vdots & \ddots & \vdots \\
    x_{G1} & x_{G2} & \cdots & x_{Gn}
\end{bmatrix}
\]

\( x_{gi} = \text{expression measure for gene } g \text{ in mRNA sample } i. \)

It is assumed that the arrays have already been conormalized.
Classifiers

A **classifier** or **predictor** $C$ for $K$ tumor classes is a map from the space $\mathcal{X}$ of gene expression profiles into the integers $\{1, 2, \ldots, K\}$.

That is, a classifier **partitions** the space $\mathcal{X}$ of gene expression profiles into $K$ disjoint and exhaustive subsets, $A_1, \ldots, A_K$, such that for a sample with expression profile $x = (x_1, \ldots, x_G) \in A_k$ the predicted class is $k$.

$$C : \mathcal{X} \to \{1, 2, \ldots, K\}.$$  

Predicted class for an observation $x \in A_k$, $\hat{y} = C(x) = k$.

---

Decision theory

Classification can be viewed as a **statistical decision theory** problem.

Assume observations are independently and identically distributed (i.i.d.) from an unknown multivariate distribution.

The class $k$ **prior**, or proportion of objects of class $k$ in the population, is denoted as $\pi_k = p(Y = k)$.

Objects in class $k$ have feature vectors with **class conditional density** $p_k(x) = p(x | Y = k)$.

---

Classifiers

Classifiers are built from past experience, i.e., from observations which are known to belong to certain classes. Such observations comprise the **learning (training) set**, $LS$

$$\mathcal{L} = \{(x_1, y_1), \ldots, (x_n, y_n)\},$$

$n_k = \#$ of class $k$ observations.

Classifier built from a learning set $\mathcal{L}$: $C(\cdot; \mathcal{L}) : \mathcal{X} \to \{1, 2, \ldots, K\}$.

Predicted class for an observation $x \in A_k$, $\hat{y} = C(x; \mathcal{L}) = k$.

The random nature of the learning set $\mathcal{L}$ implies that the prediction $\hat{y} = C(x; \mathcal{L})$ can also be viewed as **random** for a fixed value of $x$, depending on whether one conditions on the learning set or not.

---

Decision theory

When (unrealistically) both $\pi_k$ and $p_k(x)$ are known, the classification problem has a solution Bayes rule.

This situation also gives upper bounds on the performance of classifiers in the more realistic setting where these quantities are not known Bayes risk.
Decision theory

It will be useful to introduce the notion of a **loss function**. The loss function $L(h, l)$ simply elaborates the loss incurred if a class $h$ case is erroneously classified as belonging to class $l$.

The **risk function** for a classifier is the expected loss when using it to classify, that is,

$$R(C) = E[L(Y, C(X))] = \sum_k E[L(k, C(X))|Y = k] \pi_k$$

$$= \sum_k \int L(k, C(x)) p_k(x) \pi_k.$$

N.B. Here, the classifier is **fixed**, i.e., probabilities are **conditional** on the learning set. In other circumstances, the classifier may be viewed as random and $x$ as fixed.

Bayes rule

In the unlikely situation that the class conditional densities $p_k(x) = p(x|Y = k)$ and class priors $\pi_k$ are known, let

$$p(k \mid x) = \frac{\pi_k p_k(x)}{\sum_l \pi_l p_l(x)}$$

denote the **posterior probability** of class $k$ given feature vector $x$.

The **Bayes rule** predicts the class of an observation $x$ by that with highest posterior probability

$$C_B(x) = \arg\max_k p(k \mid x).$$

The Bayes rule minimizes the total risk under a symmetric loss function - **Bayes risk**.

Decision theory

Typically $L(h, h) = 0$, and in many cases the loss is symmetric, with $L(h, l) = 1, h \neq l$ making an error of one type is equivalent to making an error of a different type.

Then, the risk is simply the **misclassification rate**

$$R(C) = p(C(X) \neq Y) = \sum_k \int_{C(x) \neq k} p_k(x) \pi_k.$$

However, for some important examples such as diagnosis, the loss function is not symmetric.

Bayes rule

For a general loss function, the classification rule which minimizes the total risk is

$$C_B(x) = \arg\min_h \sum_{k=1}^K L(h, l) p(h \mid x).$$

(1)

Suitable adjustments can be made for specific loss functions and to accommodate the doubt and outlier classes.
Classifiers as estimators of the Bayes rule

Many classifiers can be viewed as versions of this general rule with particular parametric or non-parametric estimates of $p(k \mid x)$. There are two general paradigms (Friedman, 1996).

1. Direct function estimation approach. Class conditional probabilities $p(k \mid x)$ are estimated directly based on function estimation methodology such as regression.
   - Logistic regression;
   - Neural networks;
   - Classification trees;
   - Projection pursuit;
   - Nearest neighbor classifiers.

Classifiers as estimators of the Bayes rule

2. Density estimation approach. Class conditional densities $p_k(x) = p(x) \mid Y = k$ (and priors $\pi_k$) are estimated separately for each class and Bayes’ Theorem is applied to obtain estimates of $p(k \mid x)$.
   - Gaussian maximum likelihood discriminant rules, a.k.a. discriminant analysis;
   - Naïve Bayes methods, which approximate class conditional densities $p_k(x)$ by the product of their marginal densities on each feature variable;
   - Learning vector quantization;
   - Bayesian belief networks.

Maximum likelihood discriminant rule

The frequentist analogue of the Bayes rule is the **maximum likelihood (ML) discriminant rule**. For known class conditional densities $p_k(x) = p(x) \mid Y = k$, the ML rule predicts the class of an observation $x$ by that which gives the largest likelihood to $x$: $C(x) = \arg \max_k p_k(x)$.

In the case of equal class priors $\pi_k$, this amounts to maximizing the posterior class probabilities $p(k \mid x)$, i.e., the Bayes rule.

Fisher linear discriminant analysis

First applied in 1935 by M. Barnard at the suggestion of R. A. Fisher (1936), **Fisher linear discriminant analysis (FLDA)** consists of

1. finding linear combinations $x_a$ of the gene expression profiles $x = (x_1, \ldots, x_G)$ with large ratios of between-groups to within groups sums of squares **discriminant variables**;
2. predicting the class of an observation $x$ by the class whose mean vector is closest to $x$ in terms of the discriminant variables.
Linear and quadratic discriminant analysis

Linear and quadratic discriminant analysis arise as Bayes rules or maximum likelihood (ML) discriminant rules when features have Gaussian distributions within each class.

For multivariate normal class densities, i.e., for \( \mathbf{X} | Y = k \sim N(\mu_k, \Sigma_k) \), the Bayes rule is

\[
C(x) = \arg\min_k \left\{ (x - \mu_k)\Sigma_k^{-1}(x - \mu_k)' + \log |\Sigma_k| - 2\log \pi_k \right\}.
\]

In general, this is a quadratic rule — Quadratic discriminant analysis — QDA.

Linear and quadratic discriminant analysis

1. Linear discriminant analysis — LDA. When the class densities have the same covariance matrix, \( \Sigma_k = \Sigma \), the discriminant rule is based on the square of the Mahalanobis distance and is linear in \( x \) and given by

\[
C(x) = \arg\min_k (x - \mu_k)\Sigma^{-1}(x - \mu_k)'.
\]

2. Diagonal linear discriminant analysis — DLDA. When the class densities have the same diagonal covariance matrix \( \Sigma_k = \Delta = \text{diag}(\sigma_1^2, \ldots, \sigma_G^2) \), the discriminant rule is linear and given by

\[
C(x) = \arg\min_k \sum_{g=1}^{G} \frac{(x_g - \mu_{kg})^2}{\sigma_g^2}.
\]

Naive Bayes rule for Gaussian class conditional densities.

3. Nearest centroid. In this simplest case, it is assumed that \( \Sigma_k = I_G \), the \( G \times G \) identity matrix, and observations are classified on the basis of their Euclidean distance from class means \( \mu_k \).

Linear and quadratic discriminant analysis

The main quantity in the discriminant rule is

\( (x - \mu_k)\Sigma_k^{-1}(x - \mu_k)' \), the squared Mahalanobis distance from the observation \( x \) to the class \( k \) mean vector \( \mu_k \).

Thus, intuitively, the predicted class for an observation \( x \) is the class with closest mean vector \( \mu_k \), for a suitably defined distance function.

Interesting special cases are described below for homogeneous priors, i.e., \( \pi_k \) constant in \( k \).
Linear and quadratic discriminant analysis

For the sample Bayes or ML discriminant rules, the population mean vectors and covariance matrices are estimated from a learning set \( L \), by the sample mean vectors and covariance matrices, respectively: \( \mu_k = \bar{x}_k \) and \( \Sigma_k = S_k \).

For the constant covariance matrix case, the pooled estimate of the common covariance matrix is used:
\[
\hat{\Sigma} = \frac{\sum_k (n_k - 1) S_k}{n - K}.
\]

Weighted gene voting of Golub et al. (1999)

This is a minor variant on DLDA for two classes, \( k = 1, 2 \).

Sample DLDA classifies an observation \( x \) as 1 iff
\[
\sum_{g=1}^G \left( \frac{\bar{x}_{1g} - \bar{x}_{2g}}{\sigma_{1g}^2} \right) \left( x_g - \frac{(\bar{x}_{1g} + \bar{x}_{2g})}{2} \right) > 0.
\]

The discriminant function can be rewritten as \( \sum_g v_g \), where \( v_g = a_g (x_g - b_g) \), \( a_g = (\bar{x}_{1g} - \bar{x}_{2g})/\sigma_{1g}^2 \), and \( b_g = (\bar{x}_{1g} + \bar{x}_{2g})/2 \).

In Golub et al., \( a_g = (\bar{x}_{1g} - \bar{x}_{2g})/(\sigma_{1g} + \sigma_{2g}) \) ... (Wrong units).

Linear and quadratic discriminant analysis

The above simple rules may be modified easily to allow for unequal class priors.

Estimates of the priors may, in some cases, be obtained from the sample class proportions \( \hat{\pi}_k = n_k/n \).

As with any classifier explicitly estimating the Bayes rule, class posterior probabilities may be used to assess the confidence in the predictions for individual observations.

Linear and quadratic discriminant analysis

The linear discriminant rules described above are classical and widely used classification tools.

- Simple and intuitive: the predicted class of a test case is the class with the closest mean (using the Mahalanobis metric).
- Estimated Bayes rule: LDA is based on the estimated Bayes rule for Gaussian class conditional densities. Optimality when the model is true.
- Easy to implement: the partition has linear boundaries.
- Good performance in practice: in spite of a possibly high bias, the low variance of the LDA estimates of class posterior probabilities often results in low classification error (see discussion of bias-variance trade-off).
Linear and quadratic discriminant analysis

However, LDA has a number of obvious limitations which are due in part to its simplicity.

- Linear or even quadratic discriminant boundaries may not be flexible enough.
- Features may have mixture distributions within classes.
- In the case of too many features, performance may degrade rapidly due to over parameterization and high variance of parameter estimates.

Extensions to LDA

Mixture discriminant analysis, MDA. Class conditional densities are modeled as mixtures of Gaussian densities, with different mean vectors but the same covariance matrices. The EM algorithm may be used for maximum likelihood estimation of the parameters of the Gaussian components.

Automatic feature selection. Shrinkage methods can be applied to perform automatic feature selection. For example, the class mean vectors can be shrunken using matrices of between-groups and within-groups sums of squares. Parameters controlling the amount of shrinking can be selected by cross validation.

Flexible discriminant analysis, FDA. Viewing LDA as a linear regression problem, suggests considering more general non parametric forms of regression for building a more flexible classifier. FDA amounts to performing LDA on transformed responses and/or features. See logistic discrimination.

Penalized discriminant analysis, PDA. A penalized Mahalanobis distance is used to enforce smoothness of the within class covariance matrix of the features (possibly transformed as in FDA).

Logistic discrimination

For Gaussian class conditional densities with common covariance matrix (and other models)

$$\log p(k|x) - \log p(1|x) = \alpha_k + \mathbf{x}^T \beta_k.$$ 

This suggests modeling $$\log p(k|x) - \log p(1|x)$$ more generally by some parametric family of functions, say $$g_k(x; \theta)$$ (with $$g_1(x; \theta) \equiv 0$$). Estimates of the class posterior probabilities are then given by

$$\hat{p}(k|x) = \frac{\exp g_k(x; \hat{\theta})}{\sum_l \exp g_l(x; \hat{\theta})}.$$ 

Classification is done by the (estimated) Bayes rule, i.e.,

$$\hat{C}(x; \mathcal{L}) = \arg\max_k \hat{p}(k|x).$$
Logistic discrimination

In the machine learning literature the function $\exp a_k / \sum_l \exp a_l$ is known as the **softmax** function, in the earlier statistical literature, it is known as the **multiple logit** function.

In the linear case, one takes $g_k(x; \theta) = \alpha_k + x \beta_k$.

Logistic regression models are typically fit by maximum likelihood, using a Newton-Raphson algorithm known as **iteratively reweighted least squares** (IRLS).

Logistic discrimination provides a more direct way of estimating posterior probabilities and is easier to generalize than classical linear discriminant analysis (LDA and QDA), e.g. neural networks.

Nearest neighbor classifiers

Nearest neighbor methods are based on a measure of distance between observations, such as the Euclidean distance or one minus the correlation between two gene expression profiles.

The **k-nearest neighbor rule, k-NN**, due to Fix and Hodges (1951), classifies an observation $x$ as follows

1. find the $k$ observations in the learning set that are closest to $x$;
2. predict the class of $x$ by majority **vote**, i.e., choose the class that is most common among those $k$ neighbors.

Nearest neighbor classifiers: which $k$?

In general, the number of neighbors $k$ can be chosen by **cross-validation**.

Each observation in the learning set is treated in turn as if it were in the test set: its distance to all of the other learning set cases (except itself) is computed and it is classified by the nearest neighbor rule.

The classification for each learning set observation is then compared to the truth to produce the cross-validation error rate.

This is done for a number of $k$’s and the $k$ for which the cross-validation error rate is smallest is retained.
Nearest neighbor classifiers

Nearest neighbor classifiers were initially proposed by Fix and Hodges (1951) as consistent non-parametric estimates of maximum likelihood discriminant rules.

Non-parametric estimates of the class conditional densities $p_k(x)$ are obtained by first reducing the dimension of the feature space $\mathcal{X}$ from $G$ to one using a distance function.

The proportions of neighbors in each class are then used in place of the corresponding class conditional densities in the maximum likelihood discriminant rule.

Extensions of nearest neighbor rule

The nearest neighbor rule can be refined to deal with issues of unequal class priors, differential misclassification costs, and feature selection. Many of these refinements involve some form of weighted voting for the neighbors.

Class priors. Suppose the class prior probabilities $\pi_l$ are known and the learning set class proportions $\hat{\pi}_l = n_l / n$ are different from the $\pi_l$.

- Votes may be weighted according to neighbor class, $w_i = \pi_l / n_l$.

The distance to class $l$ cases may be weighted by $(n_l / \pi_l)^G$ for a $G$ dimensional feature space (Brown & Koplowitz, 1979).

Extensions of nearest neighbor rule

Distance weights. The standard $k$ nearest neighbor rule equally weights all $k$ neighbors, regardless of their distance from the test case. A more sensitive rule may be obtained by assigning weights to the neighbors that are inversely proportional to their distance from the test sample – somewhat controversial.

Differential misclassification costs. The minimum vote majority may be adjusted based on the class to be called. This corresponds to weighted voting by the neighbors, where weights are suggested by equation (1).

E.g. When $L(h, l) = L_h I(h \neq l)$, the weights are given by $w_i = L_i$.

Extensions of nearest neighbor rule

Feature selection. One of the most important issues in $k$-nearest neighbor classification is the choice of a distance function and the selection of relevant features.

For instance, in the context of gene expression data, a large number of genes are constantly expressed across classes, thus resulting in a large number of irrelevant or noise variables. Inclusion of features with little or no relevance can substantially degrade the performance of the classifier.

Feature selection may be implemented automatically, by modifying the distance function used by the classifier, or by using classification trees.
Extensions of nearest neighbor rule

**Friedman (1994).** Flexible metric nearest neighbor classification approach in which the relevance of each feature (or linear combination of features) is estimated locally for each test case. The proposed procedure, the *machette*, is a hybrid between classical nearest neighbor classifiers and tree-structured recursive partitioning techniques.

**Hastie & Tibshirani (1996).** In the *discriminant adaptive nearest neighbor* procedure, DANN, the distance function is based on local discriminant information.

---

Classification trees

**Binary tree structured classifiers** are constructed by repeated splits of subsets (nodes) of the measurement space $\mathcal{X}$ into two descendant subsets, starting with $\mathcal{X}$ itself. Each terminal subset is assigned a class label and the resulting partition of $\mathcal{X}$ corresponds to the classifier.

Three main aspects of tree construction: (i) the selection of the splits; (ii) the decision to declare a node terminal or to continue splitting; (iii) the assignment of each terminal node to a class.

Different tree classifiers use different approaches to deal with these three issues. Here, we use **CART Classification And Regression Trees** of Breiman et al. (1984).

---

Extensions of nearest neighbor rule

**Buttrey & Karo (2002).** Hybrid or composite classifier, *k-NN-in-leaf*, which partitions the feature space using a classification tree and classifies test set cases using a standard $k$-nearest neighbor rule using only training cases in the same leaf as the test case.

Each $k$ NN classifier may have a different number of neighbors $k$, a different set of feature variables, and a different choice of scaling.

The $k$-NN-in-leaf classifier differs from Friedman’s machette in that only one tree is ever grown.

---

Classification trees

1. **Splitting rule.** At each node, choose the split that maximizes the decrease in impurity.

   E.g. of **impurity functions**:
   
   Gini index $\phi(p_1, \ldots, p_k) = \sum_{k \neq i} p_i p_i = 1 - \sum_i p_i^2$, entropy, and tweeking rule.

2. **Split-stopping rule.** Grow a large tree, selectively prune the tree upward, getting a decreasing sequence of subtrees. Use **cross validation** to identify the subtree having the lowest estimated misclassification rate.

3. **Class assignment rule.** For each terminal node, choose the class that minimizes the resubstitution estimate of the misclassification probability, given that a case falls into this node.

Refinements: class priors, loss function, splits based on linear combinations of variables, missing values (surrogate splits).
Support vector machines

Support vector machines, SVM, were introduced by Vapnik (1979, 1998).

For binary classification (−1 vs. 1), the main idea is to find the best hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$ separating the classes in the learning set.

What is best? Maximize the margin, i.e., the sum of the distances from the hyperplane to the closest positive and negative correctly classified samples, while penalizing for the number of misclassifications.

One can search for the hyperplane in the original space, linear SVMs, or in a higher dimensional space, non-linear SVMs.

Support vector machines

In many situations it is useful to consider non-linear (in $\mathbf{x}$) decision functions or class boundaries. Mapping the data into higher dimensional spaces and then reducing the problem to the linear case leads to a simple solution.

Suppose the data are mapped from $\mathbb{R}^G$ to $\mathcal{H}$ using $\Phi(\cdot)$, then the training algorithm depends on the data only through the dot product $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$.

If there is a kernel function $K$ such that $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$, then one does not even need to know $\Phi(\cdot)$ explicitly.

Support vector machines

When the optimization problem is reformulated in terms of Lagrangians, the feature vectors $\mathbf{x}$ only appear in the objective function via dot products $\mathbf{x}_i \cdot \mathbf{x}_j$.

For a cost parameter $C$, building the classifier involves maximizing the dual

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

subject to $0 \leq \alpha_i \leq C$ and $\sum_i \alpha_i y_i = 0, \forall i$. The solution is given by

$$\mathbf{w} = \sum_{i=1}^{N_S} \alpha_i y_i \mathbf{x}_i,$$

where $N_S$ is the number of support vectors.

Support vector machines

User-specified parameters for the SVM C-classification method: the type of kernel $K$, i.e., non-linear transformation of the original data, and the cost parameter $C$ for misclassifications when the training data are non-separable in the transformed space.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial of degree $p$</td>
<td>$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$</td>
</tr>
<tr>
<td>Gaussian radial basis function (RBF)</td>
<td>$K(\mathbf{x}, \mathbf{y}) = \exp(-</td>
</tr>
<tr>
<td>Two layer sigmoidal neural network</td>
<td>$K(\mathbf{x}, \mathbf{y}) = \arctan(\mathbf{x} \cdot \mathbf{y} - \delta)$</td>
</tr>
</tbody>
</table>

SVMs are designed for binary outcomes. They can be generalized to multiclass problems by solving several binary problems simultaneously (see below).
Classifier partitions

The following figures compare the partitions produced by different classifiers: LDA, QDA, k NN, and CART.

The classifiers were applied to the brain tumor MD survival dataset of Pomeroy et al. (2002) using only the two genes with the largest absolute $t$ statistics.

Predicted responses “survivor” and “non survivor” are indicated by shades of red and blue, respectively.

The entire learning set of $n = 60$ samples was used to build the classifiers, the resubstitution error rate is shown below the plots.

---

Example: Linear discriminant analysis

Figure 1: Brain tumor MD survival dataset. LDA partition for the two genes with the largest absolute $t$ statistics.

Example: Quadratic discriminant analysis

Figure 2: Brain tumor MD survival dataset. QDA partition for the two genes with the largest absolute $t$ statistics.

Example: Nearest neighbor classifier

Figure 3: Brain tumor MD survival dataset, k NN partitions ($k = 1, 3, 5, 11$) for the two genes with the largest absolute $t$ statistics
Feature selection

Feature selection is one of the most important issues in classification; it is particularly relevant for microarray datasets with thousands of features, most of which are unlikely to be useful for classification purposes.

Some classifiers like trees perform automatic feature selection and are relatively insensitive to the variable selection scheme.

In contrast, standard DA and nearest neighbor classifiers do not perform feature selection; all variables, whether relevant or not, are used in building the classifier.
Explicit feature selection

One-gene-at-a-time approaches. Genes are ranked based on the value of a univariate test statistic such as: $t$ or $F$ statistic; non-parametric Wilcoxon or Kruskal-Wallis statistic; $p$-value.

Possible meta-parameters include the number of genes $G$ or a $p$-value cut off. A formal choice of these parameters may be achieved by cross validation or bootstrap procedures.

Implicit feature selection

Feature selection may also be performed implicitly by the classification rule itself.

In classification trees, features are selected at each step based on reduction in impurity and the number of features used (or size of the tree) is determined by pruning the tree using cross-validation. Thus, feature selection is an inherent part of tree building and pruning deals with the issue of over fitting.

Shrinkage methods and adaptive distance functions may be used for linear discriminant analysis and nearest neighbor classification.

Explicit feature selection

Multivariate approaches. More refined feature selection procedures consider the joint distribution of the expression measures, in order to detect genes with weak main effects but possibly strong interactions.

Bo & Jonassen (2002): subset selection procedures for screening gene pairs to be used in classification.

Breiman (1999): select genes according to importance statistics which are defined in terms of prediction accuracy (see random forests below).

Feature selection and performance assessment

The importance of taking feature selection into account when assessing the performance of the classifier cannot be stressed enough.

Feature selection is an aspect of building the classifier, whether done explicitly or implicitly.

Thus, when using for example cross validation to estimate generalization error, feature selection should be done not on the entire learning set, but separately for each cross validation sample used to build the classifier.

Leaving out feature selection from cross validation or other resampling based performance assessment method results in overly optimistic error rates.
Distance and standardization

See lecture “Distances and expression measures” for more detail.

Both the distance function and the scale the observations are measured in can have a large impact on the performance of the classifier, however, the effect varies depending on the classifier.

The choice of transformation and distance function should thus be made jointly and in conjunction with the choice of classifier.

Loss function

In many diagnosis settings, the loss incurred from misclassifying a diseased (d) person as healthy (h) far outweighs the loss incurred by making the error of classifying a healthy person as diseased.

These differential misclassification costs should be reflected in the loss function.

Suppose the loss from the first error is $e > 1$ times higher than that from the second. In the case of the Bayes rule, one could modify the posterior probability cut-offs and classify a patient as diseased if $p(d \mid x) > c = 1/(1 + e)$. Such a classifier would minimize the risk for the loss function $L(d, h) = eL(h, d)$.

Distance and standardization

Classification trees. Trees are invariant under monotone transformations of individual features (genes), e.g., standardization. They are not invariant to standardization of the observations (normalization in the microarray context).

Linear and quadratic discriminant analysis. Based on the Mahalanobis distance of the observations from the class means. Thus, the classifiers are invariant to standardization of the variables (genes), but not the observations (arrays).

Nearest neighbor classifiers. One must explicitly decide on an appropriate standardization and distance function for the problem under consideration. These classifiers are in general affected by standardization of both features and observations.

Unequal sample class representation

In many situations, such as medical diagnosis, the representation of the classes in the learning set does not reflect their importance in the problem.

For example, in a binary classification problem with a rare disease class (d) and a common healthy class (h), a learning set obtained by random sampling from the population would contain a vast majority of healthy cases.

Unequal class sample sizes could possibly lead to serious biases in the estimation of posterior probabilities $p(d \mid x)$ and $p(h \mid x)$. 
Unequal sample class representation

Consider the case of linear discriminant analysis which assumes a common covariance matrix estimated using both samples. The pooled estimate of the covariance matrix will be dominated by the more abundant sample.

This is fine if the covariance matrix is really the same for both classes. However, in the case of unequal covariance matrices, the bias in estimating class posterior probabilities \( p(k \mid x) \) is more severe when the classes are unequally represented in the learning set.

Biased sampling of classes

Downweighting and subsampling are helpful in dealing with the estimation bias for densities \( p_k(x) \), however they effectively make the sample proportions differ from the population proportions. This leads to biased estimates of the class posterior probabilities \( p(k \mid x) \).

The plug-in estimators of class posterior probabilities are in fact estimating quantities proportional to \( p(k \mid x)n_k/\pi_k \).

Adjustment is thus required to ensure that estimators of the class posterior probabilities \( p(k \mid x) \) are approximately unbiased.

This can be done by specifying appropriate priors for DA and CART, and by using weighted voting for nearest neighbors.

Biased sampling of classes

Approaches to alleviate estimation biases

- Subsample the abundant population so that both classes are on an equal footing in the parameter estimation. Problem: this would be wasteful of training data.
- Downweight cases from the abundant class so that the sum of the weights is equal to the number of cases in the less abundant class.
- Obtain less biased estimates of the class posterior probabilities, either by theory (as can be done for LDA) or via bias reduction techniques such as the jackknife.

Polychotomous classification

It may be advantageous to convert a polychotomous or \( K \) class classification problem into a series of binary or two class problems (e.g. for a large number of classes with unequal representation in the learning set).

- **All pairwise binary classification problems.** Consider all \( \binom{K}{2} \) binary classification problems; the final predicted class is the class that is selected most often as the predicted class or winner in the \( \binom{K}{2} \) decisions.
- **One-against-all binary classification problems.** Application of \( K \) binary rules to a test case \( x \) yields estimates of class probabilities; the final \( K \) class decision rule selects the class with largest estimated posterior probability.
Missing data

Some classifiers are able to handle missing values by performing automatic imputation (e.g., trees), while others either ignore missing data or require imputed data (e.g., LDA).

Classification trees can readily handle missing values through the use of surrogatesplits, i.e., splits that are most similar to the best split at a particular node.

For microarray data, simple weighted nearest neighbor imputation was found to provide accurate and robust estimates of missing values (Troyanskaya et al., 2001).

Performance assessment


   Problem: can be severely biased downward.

2. Test set estimation. In the absence of a genuine test set, cases in the learning set \( L \) may be repeatedly randomly divided into two sets, \( L_1 \) and \( L_2 \); the classifier is built using \( L_1 \) and the error rate is computed for \( L_2 \).

   Problem: reduces effective sample size; no widely accepted guidelines for choosing the relative size of these artificial learning sets and test sets.

Performance assessment

Virtually every application of classification methods to microarray data includes some discussion of the performance of the classifier(s).

The ability to predict biological outcomes using gene expression measures is usually evaluated based on estimates of the classification error.

The reported error rates are often biased downward and give an overly optimistic view of the predictive power of expression data.

Obtaining accurate or honest estimates of classification error is a very important and delicate problem.

3. \( V \)-fold cross-validation (CV) estimation.

   Cases in the learning set \( L \) are randomly divided into \( V \) subsets \( L_v, v = 1, \ldots, V \), of as nearly equal size as possible. Classifiers are built on learning sets \( L - L_v \), test set error rates are computed for \( L_v \), and averaged over \( v \).

   Bias variance trade off: small \( V \)s typically give a larger bias, but a smaller variance and mean squared error.
Performance assessment

4. **Leave-one-out cross-validation (LOOCV)**. Special case for \( V = n \).
   - In general low bias but high variance.
   - For stable (low variance) classifiers such as \( k \) NN, LOOCV provides good estimates of generalization error rates.
   - Computationally intensive for large \( n \).

5. **Out-of-bag estimation.** See discussion of random forests below.

---

Performance assessment

However, in most cases, the important features are unknown and the intended inference includes feature selection.

Then, CV estimates as above tend to suffer from a **downward bias** and inference is not warranted.

Features should be selected only on the basis of the samples in \( \mathcal{L} - \mathcal{L}_v \) for CV estimation.

---

Performance assessment

The above remarks apply to other error rate estimation methods. E.g. test set and out-of-bag estimation.

They also apply to other aspects of the classifier training process. E.g. choice of the number of neighbors \( k \) for \( k \)-NN, choice of kernel \( K \) and cost parameter \( C \) in SVMs.
Performance assessment

In the case of unequal representation of the classes, some form of stratified sampling may be needed to ensure balance across important classes in all subsamples.

In addition, for complex experimental designs, such as factorial or time course designs, the resampling mechanisms used for computational inference should reflect the design of the experiment.

Bias, variance, and error rates

The random nature of the learning set $\mathcal{L}$ implies that for any realization $x$ of the feature vector, the predicted class $\hat{Y} = \hat{C}(x; \mathcal{L})$ is a random variable.

Intuitively, for a fixed value of the feature vector $x$, as the learning set varies, so will the predicted class $\hat{Y} = C(x; \mathcal{L})$.

It is thus meaningful and instructive to consider distributional properties (e.g. bias and variance) of classifiers when assessing and comparing the performance of different classifiers.

Bias, variance, and error rates

For simplicity, consider binary classification, i.e., $K = 2$ and $Y \in \{ -1, 1 \}$, and let $f(x) = p(1|x) = p(Y = 1|X = x)$ denote the posterior probability for class 1.

Consider predicted classes $\hat{C}(x; \mathcal{L})$ based on estimates $\hat{f}(x; \mathcal{L})$ of the class posterior probabilities $f(x)$.

Denote the mean and variance of the estimators $\hat{f}(x; \mathcal{L})$ by

$$\mu_x = E[\hat{f}(x; \mathcal{L})],$$

$$\sigma_x^2 = E[(\hat{f}(x; \mathcal{L}) - \mu_x)^2].$$

Bias and variance – Estimation error

The mean squared estimation error, averaged over learning sets $\mathcal{L}$, is

$$MSE_x = E[(\hat{f}(x; \mathcal{L}) - f(x))^2] = \sigma_x^2 + (\mu_x - f(x))^2$$

$$= \text{Variance} + \text{Bias}^2. \quad (2)$$

Recall again that here $x$ is fixed and the randomness comes from the learning set $\mathcal{L}$. 
Bias and variance - Classification error

Now consider the effect of $\mu_x$ and $\sigma_x^2$ on classification error (Friedman, 1996). The Bayes rule is fixed and independent of $L$, and for binary classification problems it is given by

$$y_B = \begin{cases} 
  1, & \text{if } f(x) \geq 1/2 \\
  -1, & \text{if } f(x) < 1/2
\end{cases}.$$ 

Bias and variance - Classification error

The classification error is

$$p(\hat{Y} \neq y_B | X = x) = p(\hat{Y} = 1 | X = x) p(Y = 1 | X = x) + p(\hat{Y} = -1 | X = x) p(Y = -1 | X = x) = p(\hat{Y} = -1 | X = x) f(x) + p(\hat{Y} = 1 | X = x) (1 - f(x)) = |2f(x) - 1| p(\hat{Y} \neq y_B | X = x) + p(Y \neq y_B | X = x)$$

Estimation error + Bayes error rate.

Bias, variance, and error rates

Comparison of equations (2) and (4) shows that moments $\mu_x$ and $\sigma_x^2$ of the distribution of $f(x; L)$ have a very different impact on estimation error (for $f(x)$) and on classification error.

The bias-variance trade-off is very different for these two types of error: for estimation error, the dependence on bias and variance is additive; for classification error, there is an interaction effect.

Variance tends to dominate bias for classification error. This suggests that certain methods with high bias for function estimation may nonetheless perform well for classification because of their low variance. E.g. Naive Bayes (DLDA) and nearest neighbors.
Aggregating classifiers

Breiman (1996, 1998) found that gains in accuracy could be obtained by aggregating predictors built from perturbed versions of the learning set. In classification, the multiple versions of the predictor are aggregated by voting.

Let \( C(.; \mathcal{L}_b) \) denote the classifier built from the \( b \)th perturbed learning set \( \mathcal{L}_b \) and let \( w_b \) denote the weight given to predictions made by this classifier. The predicted class for an observation \( x \) is given by

\[
\text{argmax}_k \sum_b w_b I(C(x; \mathcal{L}_b) = k).
\]

Key to improved accuracy: instability of the classifier, i.e., variance.

Bagging

Standard bagging. In the simplest form of bagging, bootstrap replicates of the learning set, i.e., are drawn at random with replacement from the learning set.

Predictors are built for each perturbed dataset and aggregated by plurality voting (\( w_b = 1 \)).

Parametric bootstrap. Perturbed learning sets can be generated according to a mixture of multivariate Gaussian distributions.


Random forests

A random forest is a collection of tree classifiers, where each tree depends on the value of a random vector, i.i.d. for all trees in the forest (Breiman, 1999).

- Random learning set - bagging. Each tree is formed from a bootstrap sample of the learning set (the random vector consists of the outcomes of \( n \) draws at random with replacement from \( \{1, \ldots, n\} \)).

- Random features. For a fixed parameter \( G_0 \ll G \) (e.g., \( \sqrt{G} \)), \( G_0 \) features are randomly selected at each node and only these are searched through for the best split - the random vector consists of the outcomes of \( G_0 \) draws at random without replacement from \( \{1, \ldots, G\} \).

Random forests

The above two sources of randomness are combined.

A maximal exploratory tree is grown (pure terminal nodes) for each bootstrap learning set, and the forest obtains a classification by plurality voting.

Note that the main ideas in random forests are applicable to other types of classifiers than trees.
I. **Out-of-bag estimate of error rate.** No need for CV or test set estimation of error rate; an unbiased estimate is obtained as a by-product of the bootstrap.

For each bootstrap sample, about 1/3 of the cases are left out and not used in the construction of the tree ⇒ out-of-bag test set.

For the $b$th bootstrap sample, put the out-of-bag cases down the $b$th tree to get a test set classification. Let the final test set classification of the forest be the class having the most votes. Compare this classification to the class labels of the learning set to get the out of bag estimate of the error rate.

II. **Case-wise information.**

- **Prediction votes.** The proportions of votes for each class can be viewed as estimates of the class posterior probabilities ($\in [0,1]$).
  The prediction vote for the winning class gives a measure of confidence for the prediction of individual observations.

- **Vote margins.** Vote margins are defined as the proportion of votes for the true class minus the maximum of the proportions of votes for each of the other classes ($\in [-1,1]$). Low vote margins may indicate that the corresponding samples are hard to predict or, in some cases, they may reflect mislabeled learning set cases.

---

III. **Variable importance statistics.** Here, variable importance is defined in terms of the contribution to prediction accuracy, i.e., predictive power.

For each tree, randomly permute the values of the $g$th variable for the out-of-bag cases, put these new covariates down the tree, and get new classifications for the forest.

The importance of the $g$th variable can be defined in a number of ways.

*Importance measure 1:* the difference between the out of bag error rate for randomly permuted $g$th variable and the original out-of-bag error rate.

*Importance measure 2:* the average across all cases of the differences between the margins for the randomly permuted $g$th variable and for the original data.

*Importance measure 3:* the number of lowered margins minus the number of raised margins.

*Importance measure 4:* the sum of all decreases in impurity in the forest due to the $g$th variable, normalized by number of trees.
By-products of aggregation

- IV. Intrinsic proximities between cases. Proportion of trees for which cases $i$ and $j$ are in the same terminal node.
  - Clustering;
  - Multidimensional scaling;
  - Outlier detection: $1/(\text{sum of squared proximities of cases in same class}).$

---

**Boosting**

The data are resampled adaptively so that the weights in the resampling are increased for those cases most often misclassified.

The aggregation of predictors is done by weighted voting.


---

**Comparison study – classifiers**

- Linear and quadratic discriminant analysis
  - Fisher linear discriminant analysis (FLDA);
  - Diagonal linear discriminant analysis (DLDA)
  - Gene voting scheme of Golub et al. is a variant of DLDA;
  - Diagonal quadratic discriminant analysis (DQDA).
- Nearest neighbor classifiers.
- Classification trees (CART).
- SVMs.
- Bagging, boosting, random forests.

N.B. DLDA and DQDA are a.k.a. naive Bayes classifiers.

Comparison study - datasets

- **Lymphoma.** Alizadeh et al. (2000) (cDNA arrays).
  
  \( n = 81 \) samples; \( G = 4,682 \) genes, 3 classes (B-CLL, FL, DLBCL).

- **Leukemia.** Golub et al. (1999) (Affymetrix chips).
  
  \( n = 72 \) samples; \( G = 3,571 \) genes, 3 classes (B cell ALL, T cell ALL, AML).

- **NCI 60.** Ross et al. (2000) (cDNA arrays).
  
  \( n = 64 \) samples; \( G = 5,244 \) genes, 8 classes.

- **Brain cancer.** Pomeroy et al. (2002) (Affymetrix chips).
  
  Classic vs. desmoplastic MD: \( n = 34 \) samples, \( G = 5,898 \) genes,
  
  Survivors vs. non survivors: \( n = 60 \) samples, \( G = 4,459 \) genes.

- **Breast cancer.** West et al. (2001) (Affymetrix chips).
  
  ER +ve vs. ER -ve: \( n = 49 \) samples, \( G = 7,129 \) genes,
  
  Nodal +ve vs. nodal -ve: \( n = 49 \) samples, \( G = 7,129 \) genes.

Comparison study - results

- Simple classifiers such as naive Bayes (DLDA) and k-NN performed remarkably well compared to more complex ones.

- Aggregation improved the performance of unstable classifiers such as CART.

- The impact of gene screening varied depending on the dataset and classifier. Some screening of the genes down to \( G = 10 \) – 100 seems advisable.

- It is essential to take into account gene screening and other training decisions in error rate estimation procedures.

Discussion

- As with multiple testing, the strong interest in classification in microarray experiments has resulted in:
  
  \( \rightarrow \) lots of papers;
  
  \( \rightarrow \) old methods with new names;
  
  \( \rightarrow \) new methods with inadequate or unknown properties;
  
  \( \rightarrow \) a lot of confusion!

- New proposals should be formulated precisely, within the standard statistical framework, to allow a clear assessment of the properties of different procedures.

- Honest performance assessment (see also West et al. (2001)).
Discussion

There are two main goals

- **Prediction.** Predict biological outcomes for future samples (tumor class, survival) using a collection of predictor variables (gene expression measures).
- **Information.** Extract information about the underlying data generating mechanism, i.e., on the relationship between responses and predictor variables.

---

**R classification software**

- `class`: knn and lvq functions.
- `e1071`: svm function.
- `ipred`: baggin, resampling based estimation of prediction error.
- `LogitBoost`: boosting for tree stumps.
- `MASS`: lda, polr, and qda functions.
- `RanForests`: http://berkeley.edu/users/breiman/.
- `rpart`: classification and regression trees.

---

**References**


**Data models**: High dimensional data, unknown distribution, many models will provide similar predictive accuracy.

**Accuracy vs. simplicity (interpretability)**: a predictor does not have to be simple to provide accurate prediction or reliable information about the relationship between responses and predictor variables (e.g. random forests).

**Dimensionality**: newer classification procedures thrive on the number of variables, the more the better.

Ref. Breiman (2001)
Acknowledgments

Leo Breiman, Statistics, Berkeley.

Brown Lab, Biochemistry, Stanford.

Sabina Chiaretti, Dana Farber Cancer Institute.