GS01 0163
Analysis of Microarray Data

Keith Baggerly and Kevin Coombes
Section of Bioinformatics
Department of Biostatistics and Applied Mathematics
UT M. D. Anderson Cancer Center
`kabagg@mdanderson.org`
`kcoombes@mdanderson.org`

9 November 2004

Lecture 20: Classification I

- Clustering and Classification
- Classification Methods
- LDA, DLDA, QDA
- KNN
- Validation

Discovery and Prediction

Class Discovery is most closely associated with clustering; we are using structure inherent in the data to suggest groupings of interest.

Class Prediction, by contrast, is most closely associated with classification. Here, we start with a few samples from each of a few classes known to be of interest *a priori*, and try to allocate new observations to these classes.

How Classification Works

In general, the first step in classification with microarrays is to select a subset of genes to work with, since the entire set can be problematic.

Given a set of features, various methods are then used to divide up the space of possible values into regions that are “class 1”, “class 2” and so on. Typically, these regions will be divided by smooth curves defining a “decision boundary”.

Smooth boundaries have the advantage that they can suggest some biology.

Some Classification Methods

Linear Discriminant Analysis (LDA, aka Fisher’s LDA)

Quadratic Discriminant Analysis (QDA)

Diagonal Linear Discriminant Analysis (DLDA)

Classification and Regression Trees (CART)

k Nearest Neighbors (KNN)

Support Vector Machines (SVM)

A Data Set

Assessing the importance of BRCA1 and BRCA2 mutations in breast cancer. Initially introduced in Hedenfalk et al (2001), and filtered a bit in Simon et al (2003).

texttt http://linus.nci.nih.gov/BRB-ArrayTools.html

under “Book” and “BRCA”.

Log ratio measurements on 3226 genes for 22 breast tumors, 7 with BRCA mutations and 8 with BRCA2 mutations.

Focus on dividing BRCA2 status groups.

Choosing a Subset of Genes

Most commonly, we pick those that show good univariate performance at separating the groups of interest, either by t-tests or Wilcoxon tests (2 groups) or ANOVA or Kruskal-Wallis tests (3 or more groups).

Choosing a Subset of Genes

Most commonly, we pick those that show good univariate performance at separating the groups of interest, either by t-tests or Wilcoxon tests (2 groups) or ANOVA or Kruskal-Wallis tests (3 or more groups).

This makes a pretty strong assumption that there will not be any really useful interaction effects in the absence of important information from the individual components.

What we got Here

Here, we used pooled two-sample t-tests to contrast the 8 BRCA2 samples with the 14 others, and chose to focus on just the ones that had a p-value < 0.001 (there are 49 of these).

Telling Groups Apart

In telling two groups apart, a natural starting point is to use as simple a rule as possible – drawing a straight line, mapping every observation down onto that line, and cutting the line at some central point.

Conversely, we can think of this as taking the space and using a sheet to cut it in half – everything on the left side of the sheet will be classed in group 1, and everything on the right will be classed in group 2. This is a simple decision boundary.

So, how do we choose the best line to use?

Fisher’s Linear Discriminant Analysis

In 1d, the problem is pretty straightforward – we find the two group means and place our cut precisely at the midpoint.

Fisher’s Linear Discriminant Analysis

In 1d, the problem is pretty straightforward – we find the two group means and place our cut precisely at the midpoint.

In 2d, this is still partially true – the optimal line to use is the one that connects the two group means, and the cut point is still at the middle. The question is one of how to map points in the plane down onto this line.

A Simple Example

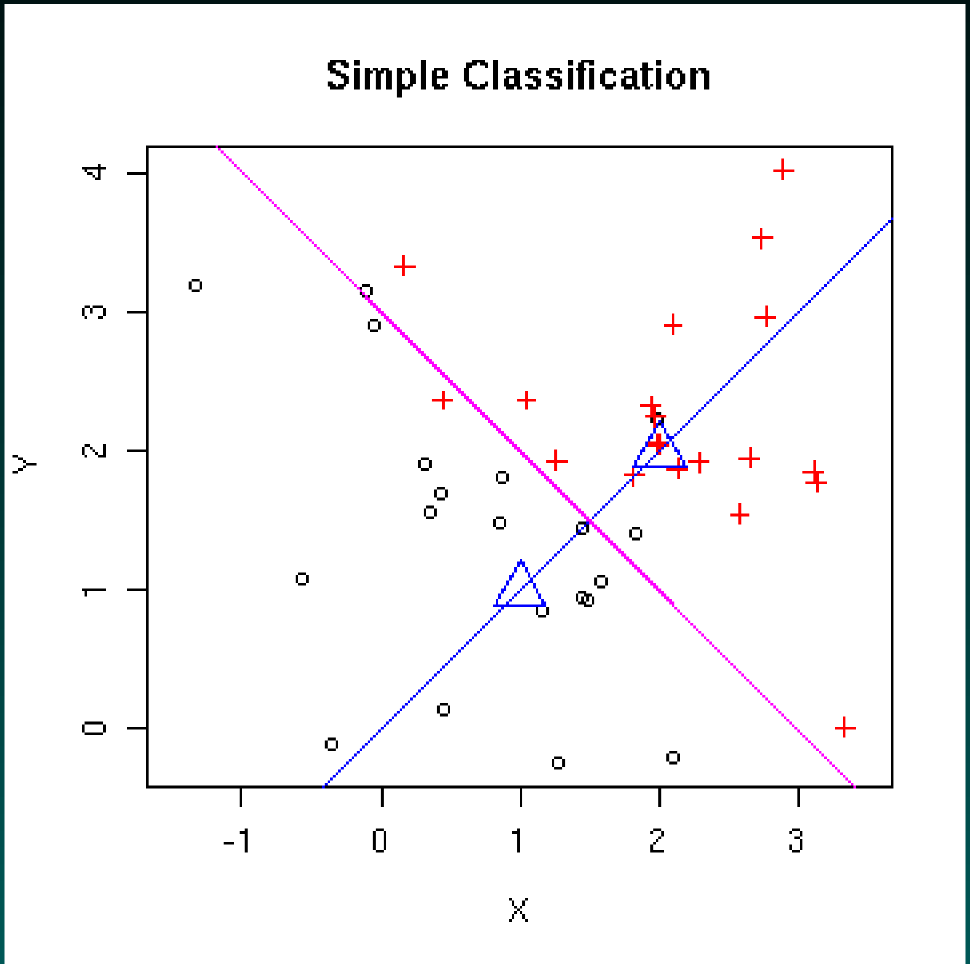
Say that the two groups have centers at (1,1) and (2,2), and that the variances are the same on the two axes. Then

the line joining the two is $y = x$,

the center point is (1.5,1.5),

and the best separating plane is orthogonal to the connecting line and passes through the center – $y = 3 - x$.

Simple Classification 1



Looks pretty easy...

Fisher's Linear Discriminant Analysis

Now say that something got screwed up, and the measurements on the x axis were sent in mm as opposed to m.

The centers are now (1000,1) and (2000,2), the central cut point is now (1500,1.5), and the connecting line is $y = x/1000$.

The optimal separating plane, however, is *not* orthogonal to this line.

If it were, the line would be $y = 1500001.5 - 1000x$.

Fisher's Linear Discriminant Analysis

But all I did was stretch the x-axis from what it was before; the optimal separator should also stretch.

Since this separator hit the y axis at 3 before the stretch, it should still do so after. Thus, the new "best line" is $y = 3 - x/1000$.

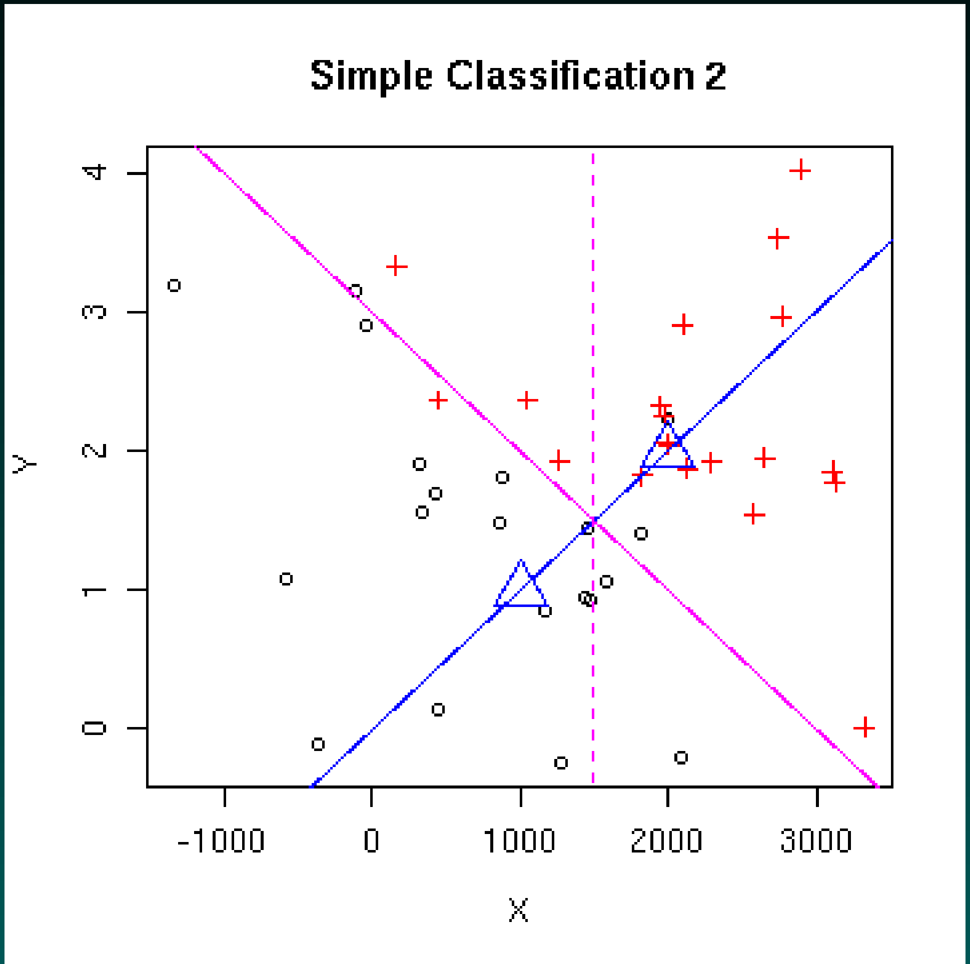
Fisher’s Linear Discriminant Analysis

But all I did was stretch the x-axis from what it was before; the optimal separator should also stretch.

Since this separator hit the y axis at 3 before the stretch, it should still do so after. Thus, the new “best line” is $y = 3 - x/1000$.

The optimal separating line should not change if we simply change our measuring units, so we need a method that is scale-invariant.

Simple Classification 2



Hmm.

Extending the t test

The t test is scale-invariant:

$$\frac{\bar{x}_1 - \bar{x}_2}{s_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

as is its square

$$(\bar{x}_1 - \bar{x}_2)\left\{s_p^2(\frac{1}{n_1} + \frac{1}{n_2})\right\}^{-1}(\bar{x}_1 - \bar{x}_2).$$

So we need to extend this by standardizing the data in general.

The Extension

Let

$$W = (\bar{x}_1 - \bar{x}_2) S^{-1} \left\{ x - \frac{1}{2}(\bar{x}_1 + \bar{x}_2) \right\}.$$

If $W > 0$, then x belongs to group 1, else it belongs to group 2.

We have the dividing line, a squashing of the space to standardize things, and a measurement along this line.

Some R Code

```
library(’MASS’);

ds1 <- matrix(rnorm(40),20,2) + 1;
ds2 <- matrix(rnorm(40),20,2) + 2;

my.lda <- lda(rbind(ds1,ds2),grouping=
      as.factor(c(rep(1,20),rep(2,20))))

my.predictions <- predict(my.lda,
                          rbind(ds1,ds2));
```

Does this Work?

Well, it certainly did for the problems Fisher used it for back in 1936. For microarray data, however, there can be some problems.

Does this Work?

Well, it certainly did for the problems Fisher used it for back in 1936. For microarray data, however, there can be some problems.

Specifically, in order to use linear discriminant analysis, we need to compute the inverse of the sample covariance matrix, and if we have k variables that means that we are estimating $k(k+1)/2$ different variances, in addition to k means. That's a lot, and can get unstable if k^2 is an appreciable fraction of the total number of samples.

Does it Work Here?

Here, when we began, $n = 22$ and $k = 49$. Whoops. Inverting the matrix will not work at all; we have to restrict our attention to a smaller number of features (say 5).

Fixum Addum Hoccum

What if we don’t use the entire covariance matrix, but rather just the main diagonal?

Fixum Addum Hoccum

What if we don’t use the entire covariance matrix, but rather just the main diagonal?

This is much better behaved for microarray data.

Fixum Addum Hoccum

What if we don’t use the entire covariance matrix, but rather just the main diagonal?

This is much better behaved for microarray data.

This approach is known as Diagonal Linear Discriminant Analysis (DLDA).

More R Code

```
d1bar <- apply(ds1,2,mean);
d2bar <- apply(ds2,2,mean);
d1var <- apply(ds1,2,var);
d2var <- apply(ds2,2,var);
dvar <- ((20-1)*d1var +
         (20-1)*d2var)/38;

W <- (d1bar - d2bar) %*%
      diag(dvar) %*%
      t(t(rbind(ds1,ds2)) -
         (d1bar + d2bar)/2);
```

Further Extensions

So far, we’ve used means and covariance matrices, but we’ve assumed that the covariances are the same for the two groups.

If we assume that the two groups are both normally distributed, but that the two covariance matrices can be different, then the contours that get drawn result from drawing concentric ellipsoids about the two group centers, and the decision boundary can be curved. This is known as Quadratic Discriminant Analysis (QDA).

And More R Code

```
library(’MASS’);

ds1 <- matrix(rnorm(40),20,2) + 1;
ds2 <- matrix(rnorm(40),20,2) + 2;

my.qda <- qda(rbind(ds1,ds2),grouping=
      as.factor(c(rep(1,20),rep(2,20))))

my.predictions <- predict(my.qda,
                          rbind(ds1,ds2));
```

Back away from the Covariance Matrix

and nobody will get hurt. Easy now...

Back away from the Covariance Matrix

and nobody will get hurt. Easy now...

The discriminant methods that we have described so far are focused on mean or central behavior.

Back away from the Covariance Matrix

and nobody will get hurt. Easy now...

The discriminant methods that we have described so far are focused on mean or central behavior.

We can also carry over some ideas from the clustering/linkage realm, and decide to classify a new sample on the basis of the classification that holds for the known samples closest to it. If we look at the k nearest neighbors (KNN), then the sample is classified according to majority vote amongst the k.

Something Odd about the Neighbors..

KNN is most explicitly defined in terms of both training and test sets. It is very rarely discussed solely in terms of the training sets which allow us to partition the space into “group 1 regions” and “group 2 regions”.

This can be more flexible, and can produce some odd decision boundaries (a mixed blessing).

And More R Code

```
library(’class’);

ds1 <- matrix(rnorm(40),20,2) + 1;
ds2 <- matrix(rnorm(40),20,2) + 2;

my.knn <- knn(
  train = rbind(ds1[1:10,],ds2[1:10,]),
  test = rbind(ds1[11:20,],ds2[11:20,]),
  k = 1,
  cl = as.factor(c(rep(1,20),rep(2,20))))
```

How Good is the Rule?

How well does it classify the data?

How Good is the Rule?

How well does it classify the data?

Is this a valid test of the classifier?

How Good is the Rule?

How well does it classify the data?

Is this a valid test of the classifier?

Probably not. If the data to be predicted is used to train the classifier in the first place, then our results will look better than they should. This is the problem of “overfitting” the data.

Our Task

We need to think of a way to assess the prediction accuracy of the rule using samples that the rule hasn’t seen.

So, where will these samples come from?

Cross-Validation

We can make use of the samples we have by using only some of them to fit the model, and then predicting the status of the one(s) we haven't seen.

We're going to do this in stages, working first in a context where there should be nothing going on to try to highlight the important issues a bit more clearly.

Working with Noise

```
# start with the null matrix
nothing.here <- matrix(rnorm(20000),
                       1000,20);

# Pick the best 5 t-test "genes"
ds1 <- nothing.here[,1:10];
ds2 <- nothing.here[,11:20];
mu1 <- apply(ds1,1,mean);
mu2 <- apply(ds2,1,mean);
var1 <- apply(ds1,1,var);
var2 <- apply(ds2,1,var);
varpool <- ((10-1)*var1+(10-1)*var2)/18;
```

Fitting the Best

```
nothing.t <- (mu1 - mu2) /
        sqrt(varpool * (1/10 + 1/10));
nothing.ranks <- rank(abs(nothing.t));
nothing.best5 <-
   nothing.here[nothing.ranks > 995,];

# Use LDA to separate the data and to
# predict the status of the 20 samples.
nothing.lda1 <- lda(t(nothing.best5),
  grouping=as.factor(
    c(rep(1,10),rep(2,10))));
```

Counting Successes

```
nothing.predictions1 <- predict(
  nothing.lda1,t(nothing.best5));
nothing.right1 <- sum(
  nothing.predictions1$class[1:10] == 1) +
                  sum(
  nothing.predictions1$class[11:20] == 2);
```

Here, we got 20 right. That looks a bit too good. How are we overfitting?

Leaving one Out

```
# Next, use cross-validation with the 5
# genes chosen above to fit the divider
# on 19, and to predict the status of
# the last one.

groupvec <- as.factor(c(rep(1,10),
                      rep(2,10)));
predvec <- rep(0,20);
for(i1 in 1:20){
  nothing.lda2 <- lda(
    t(nothing.best5[,-i1]),
    grouping=groupvec[-i1]);
```

Making the Predictions

```
  nothing.predictions2 <- predict(
    nothing.lda2,t(nothing.best5[,i1]));
    predvec[i1] <-
      nothing.predictions2$class;
}
nothing.right2 <-
  sum(predvec[1:10] == 1) +
  sum(predvec[11:20] == 2);
```

Making the Predictions

```
  nothing.predictions2 <- predict(
    nothing.lda2,t(nothing.best5[,i1]));
    predvec[i1] <-
      nothing.predictions2$class;
}
nothing.right2 <-
  sum(predvec[1:10] == 1) +
  sum(predvec[11:20] == 2);
```

Doing this, we get 17 right. Still a bit high. How are we overfitting?

Refitting Everything I

```
# Next, use cross-validation by using
# 19 samples, defining the 5 best on
# the basis of those 19, and then
# predicting the status of the last case.

predvec3 <- rep(0,20);
for(i1 in 1:10){
  mu1 <- apply(ds1[,-i1],1,mean);
  mu2 <- apply(ds2,1,mean);
```

Refit the selection of the 5 genes to be used!

Refitting Everything II

```
var1 <- apply(ds1[,-i1],1,var);
var2 <- apply(ds2,1,var);
varpool <- ((9-1)*var1 + (10-1)*var2)/17;

nothing.t <- (mu1 - mu2) /
  sqrt(varpool * (1/9 + 1/10));
nothing.ranks <- rank(abs(nothing.t));
nothing.best5 <- nothing.here[
  nothing.ranks > 995,];
```

Refitting Everything III

```
  nothing.lda3 <- lda(
    t(nothing.best5[,-i1]),
    grouping=groupvec[-i1]);
  nothing.predictions3 <- predict(
    nothing.lda3,t(nothing.best5[,i1]));
  predvec3[i1] <-
    nothing.predictions3$class;
}
```

Refitting Everything III

```
  nothing.lda3 <- lda(
    t(nothing.best5[,-i1]),
    grouping=groupvec[-i1]);
  nothing.predictions3 <- predict(
    nothing.lda3,t(nothing.best5[,i1]));
  predvec3[i1] <-
    nothing.predictions3$class;
}
```

Here, we get 6 right.

Is This Consistent?

nothing.right1: 20,20,20,etc

nothing.right2: 17,18,20,etc

nothing.right3: 6,12,8,etc