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

11 November 2004

Lecture 21: Classification II

- CART
- Genetic Algorithms
- Validation of Significance

Classification and Regression Trees (CART)

How does CART work?

CART assumes that the axes (genes) have inherent meaning, and tries to work with them directly as opposed to forming linear combinations.

This has some potential advantages in terms of interpretation, and in terms of specifying a rule.

CART splits the data using a series of binary decisions.

CART Questions

How should a split be chosen?

When should we stop splitting?

When we reach a terminal node (a leaf), what class should we say we've found?

Choosing a Split

Before we've done any splitting of the data, we have a mixture of cases and controls. We can view this as the root node, and initially we would say that this node has a certain amount of "impurity" – a node is said to be pure if all of the samples at that node are of the same class.

We want to

define a measure of impurity

find splits that reduce this measure

Defining Impurity

There are a few different mathematical ways of defining the impurity of a node; the two most common are

The entropy or information impurity:

$$-\sum_{classes} P(class) * \log_2(P(class))$$

The Gini index impurity:

$$1 - \sum_{classes} P(class)^2$$

Properties of Impurity

Both of these are peaked in the center – nodes that are split half and half are highly impure.

Similarly, both of these are 0 at the ends – nodes that are all of one class have no impurity.

Working with Gini

In the BRCA example, we compute the overall impurity by computing the impurity of the root node and multiplying it by the number of samples at that node. Here, this becomes

$$22 * \left\{ 1 - \left(\frac{8}{22} \right)^2 - \left(\frac{14}{22} \right)^2 \right\} = 10.18182.$$

Working with Gini

Now let's say that we can split this node according to the values of variable x_1. There are 13 samples with $x_1 < 0$, and all 13 of these have no BRCA2 mutations. There are 9 samples with $x_1 >= 0$, and 8 of these have BRCA2 mutations.

What is the impurity after this split?

Working with Gini Nodes

Here, we have to compute the values at the two nodes and combine them:

$$13 * \left\{ 1 - \left(\frac{13}{13} \right)^2 - \left(\frac{0}{13} \right)^2 \right\} +$$
$$9 * \left\{ 1 - \left(\frac{8}{9} \right)^2 - \left(\frac{1}{9} \right)^2 \right\} = 1.77778.$$

The reduction in impurity that we get by making this split is $10.18182 - 1.77778 = 8.40404$.

Observations about Splitting

In general, we choose the split giving the biggest overall reduction in impurity.

It's easier to split large nodes, as even small reductions in impurity are magnified by the number of samples involved.

At some point, however, it's not worth it anymore.

If we keep splitting the data until every node is completely pure, then in general we will have overfit the data. We want our splits to correspond to things we think are most likely to persist.

Ways to Stop Splitting

We can choose to stop if

The best reduction in impurity that we can get is below a certain threshold value.

The number of samples at a node gets below a specified threshold value.

Specifying these thresholds is something of a black art.

Some R Code

CART can also be viewed as *recursive partitioning*, and R uses the function `rpart`.

```
brcaSampleInfo$BRCA2
> brcaSampleInfo$BRCA2
 [1] - - - - - - + + + + -
[12] - - - - - - - + + + +
Levels: + -
brcaNumbersShort <- brcaNumbers[
   pvalsBRCA2 < 0.001,];

library('rpart');
```

Output I

```
treefit1 <- rpart(brcaSampleInfo$BRCA2 ~ .,$
      data.frame(t(brcaNumbersShort)))
> treefit1
n= 22
node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 22 8 - (0.3636364 0.6363636)
  2) X914< -0.2607988 9 1 +
      (0.8888889 0.1111111) *
  3) X914>=-0.2607988 13 0 -
      (0.0000000 1.0000000) *
```

Output II

```
> summary(treefit1)
  n= 22

     CP nsplit rel error xerror      xstd
1 0.875      0     1.000   1.00 0.2820380
2 0.010      1     0.125   1.25 0.2919371

Node number 1: 22 observations,    complexit
  predicted class=-  expected loss=0.3636364
    class counts:     8    14
   probabilities: 0.364 0.636
```


Output III (Split Not Unique?)

```
left son=2 (9 obs) right son=3 (13 obs)
Primary splits:
      X914  < -0.2607988 to the left,
        improve=8.404040, (0 missing)
      X2456 < 0.1496223  to the right,
        improve=8.404040, (0 missing)
      X2804 < 0.02940068 to the left,
        improve=8.404040, (0 missing)
      X35   < 0.5606404  to the right,
        improve=8.315152, (0 missing)
      X501  < 1.533854   to the right,
        improve=8.315152, (0 missing)
```

Output IV

```
Surrogate splits:
     X2977 < 0.1242674  to the left,
      agree=0.955, adj=0.889, (0 split)
     X35   < 0.5606404  to the right,
      agree=0.909, adj=0.778, (0 split)
     X501  < 1.533854   to the right,
      agree=0.909, adj=0.778, (0 split)
     X952  < -0.2179817 to the left,
      agree=0.909, adj=0.778, (0 split)
     X1656 < 1.052772   to the left,
      agree=0.909, adj=0.778, (0 split)
```

Output V

```
Node number 2: 9 observations
  predicted class=+  expected loss=0.1111111
    class counts:     8     1
   probabilities: 0.889 0.111

Node number 3: 13 observations
  predicted class=-  expected loss=0
    class counts:     0    13
   probabilities: 0.000 1.000
```

So, does CART work?

Unfortunately, it doesn't work all that well for most microarray data experiments.

The problem is simply that by focusing so intently on a small number of variables (dealing with ties?) that CART can get misled by random chance splits. This is less of a problem if the number of arrays in the experiment is large (50 or more) such that we are unlikely to see very large reductions in impurity even when we start with 1000s of genes.

Cross-validation shows problems.

Can we fix CART?

By averaging the predictions from several models, we may come up with more robust algorithms. The problem is that by averaging, we are combining the results from many different genes, and the simplicity of interpretation is somewhat lost.

Useful if we start with a small number of variables.

Genetic Algorithms (GAs)

Yet one more method for classification.

Ironically (for this course), GAs do not involve specific genes. Rather, the key idea is to develop a good classifier through *evolution*. This process is assumed to mimic the way in which genes evolved and gained functionality.

GAs work by specifying an objective function (such as the fraction of samples correctly classified), and trying combinations of elements (logical chromosomes) to see how well they optimize the objective.

A Simple Example

Say we want to maximize $f(x) = x^2(1-x)^3$ over the interval $[0, 1]$. We can of course solve for this analytically; the maximum value of 0.03456 is attained when $x = 0.4$, but this is a toy problem.

We want to represent the number x in terms of binary pieces, as a "logical chromosome".

Say we start with a sequence of 30 0's and 1's:

00101 10101 01010 10011 11011 01011

Treating this as a binary integer divided by 2^{30} gives 0.177089.

Testing the Fit

Here, our logical chromosome has "fitness"

$$f(0.177089) = 0.01747504.$$

This is just one random string. Let's say we generate 100 such strings. Then we can compute the fitness for each of them.

For Example...

```
> startgen[1,]
 [1] 11011 11010 11000 00111 00001 00111
> startgen[2,]
 [1] 01101 01000 00101 00000 00101 00010
> startgen[3,]
 [1] 11010 00100 00100 11101 00100 11110

> x.vals[1:3]
[1] 0.8698798 0.4142152 0.8165561

> x.fitness[1:3]
[1] 0.001667067 0.034487868 0.004116060
```

How do we Evolve?

At each generation, we pick pairs of elements to "breed", with the chance of being selected being linked to the overall fitness in some way.

Given a pair, we line them up, and let them "cross over" at some randomly chosen location. So

11011 11010 11000 00111 00001 00111
01101 01000 00101 00000 00101 00010

might produce

11011 11010 11000 00000 00101 00010

Other Tweaks

Crossover is typically the main evolutionary driver over several generations. However, there is typically also a small chance of something new getting introduced via mutation – with a small probability, a random element of the logical chromosome will be “flipped”. So

11011 11010 11000 00111 00001 00111

might produce

11011 11010 11000 10111 00001 00111

Trying it here

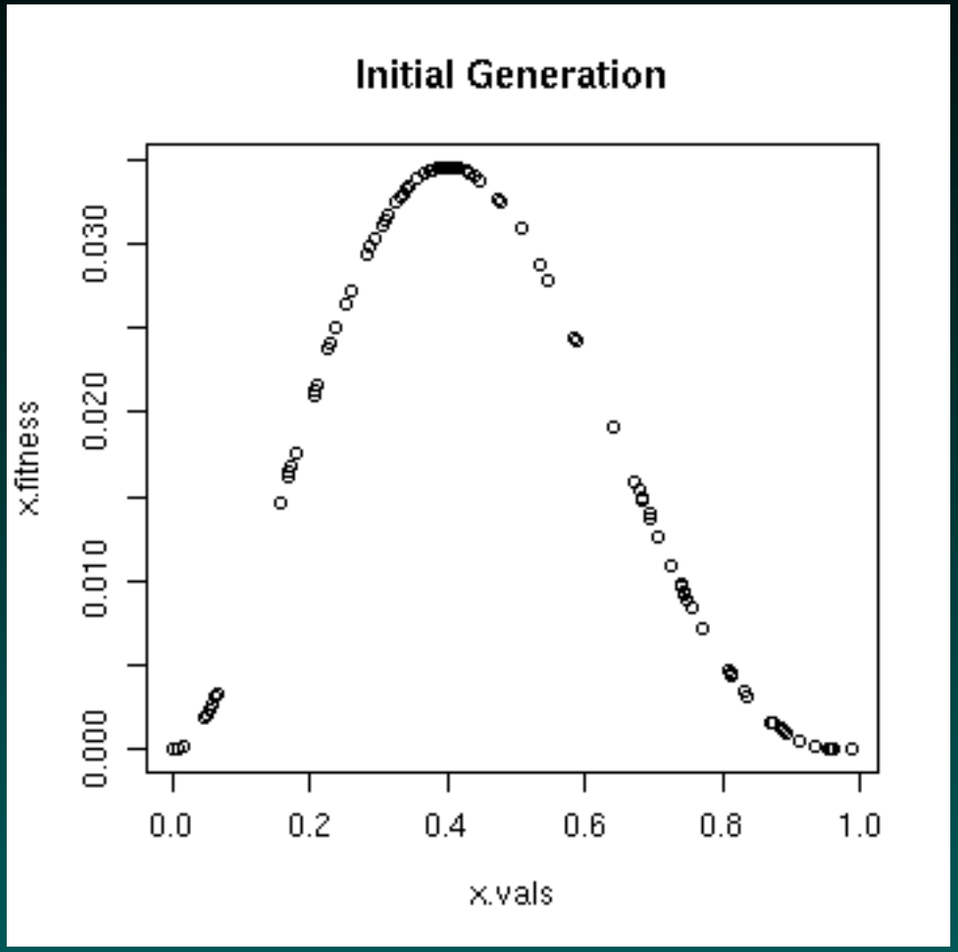
I chose a population of 100 strings of length 30.

The chance of being selected as a member of a pair for forming a next generation individual was proportional to

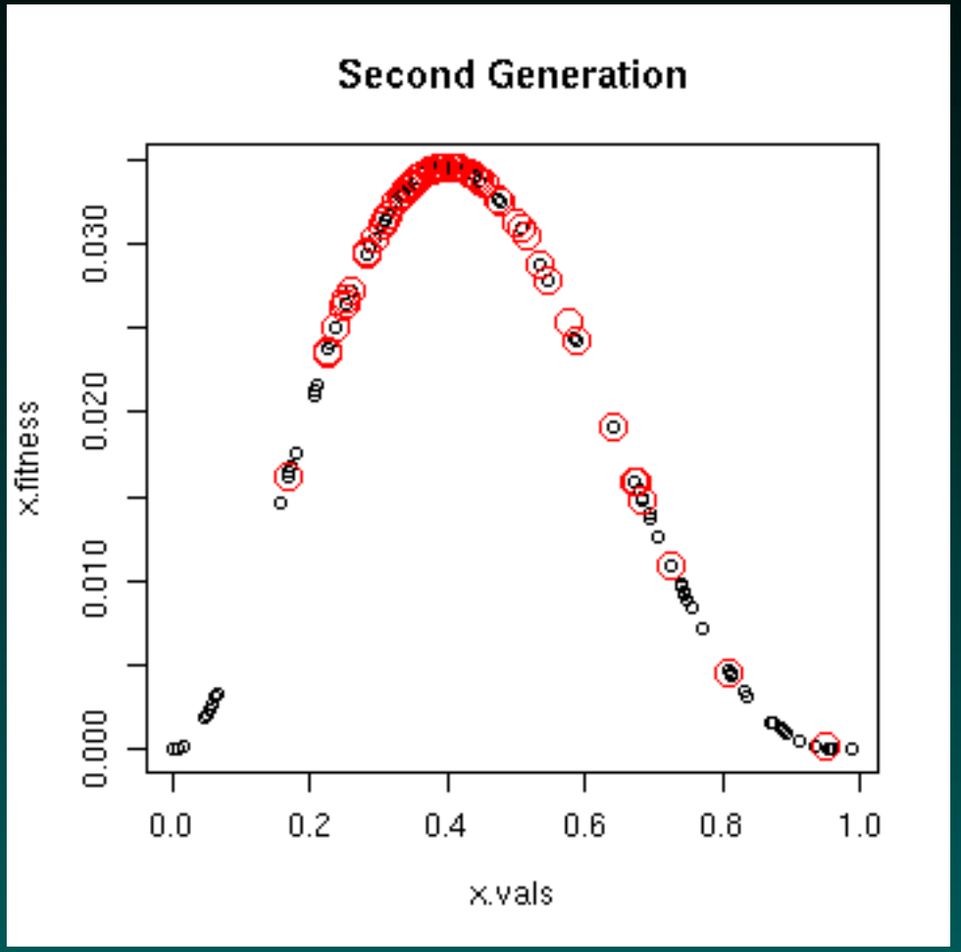
$$\left(\frac{fit(i) - \min(fit)}{\max(fit) - \min(fit)} \right)^2$$

I didn't bother with mutation, and I let things go for 10 generations.

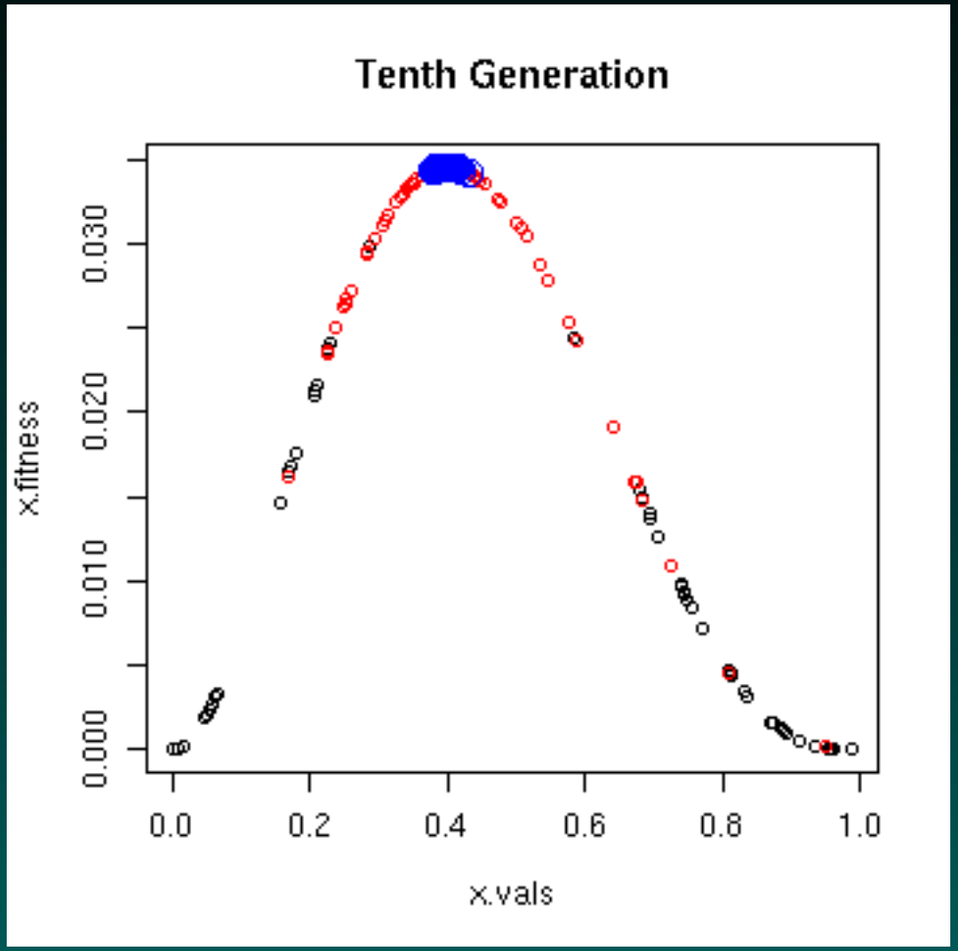
GA Fits



GA Fits



GA Fits



What Does This Gain Us?

A GA is a stochastic (random) search method. It tries lots of combinations of things, including some that might not occur to us. It has the potential of including some types of interactions that might not have otherwise been spotted.

Coupled with Arrays

In the microarray context, the logical chromosome can be used to indicate the presence or absence of a gene.

Alternatively, if we want to work with just a small number of genes (say 5), the individual "chromosomes" can be lists of 5 index values.

For each chromosome, we can compute the overall cross-validated classification accuracy using LDA, or the distance between the group centers after standardizing.

We've Used This Approach

In looking at proteomic data. Proteomic spectra yield peaks at specific m/z (loosely mass) values. Different mass peaks ideally correspond to different proteins.

We started with a 506 by 41 matrix of peak intensities (log transformed) – 506 m/z values, and 41 samples: 24 from patients with lung cancer, and 17 controls.

We then looked for good sets of 1 through 5 peaks.

What we Did

We used Mahalanobis distance as our fitness function:

$$MD = (\bar{x}_1 - \bar{x}_2)S^{-1}(\bar{x}_1 - \bar{x}_2)$$

This is the multivariate generalization of the two sample t-test.

Searching the Space

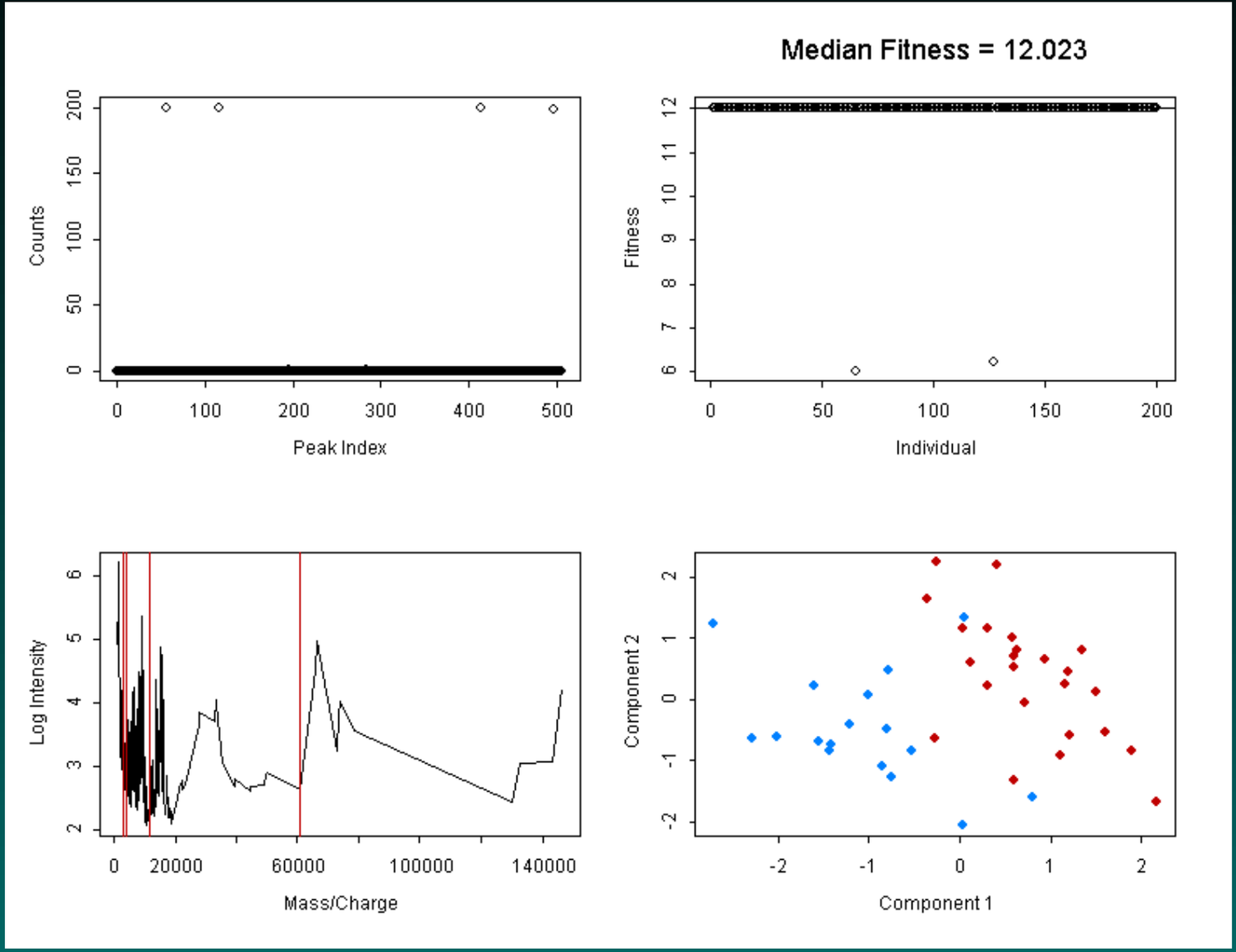
How we searched

- For 1 and 2 peaks, exhaustive search.
- For sets of 3, 4 and 5 peaks, use a Genetic Algorithm (GA).

Some GA details

- 200 logical chromosomes/run, 250 generations.
- 50 different random starts (3,4,5).
- Every run of the GA converged.

A Typical Solution: Best 4



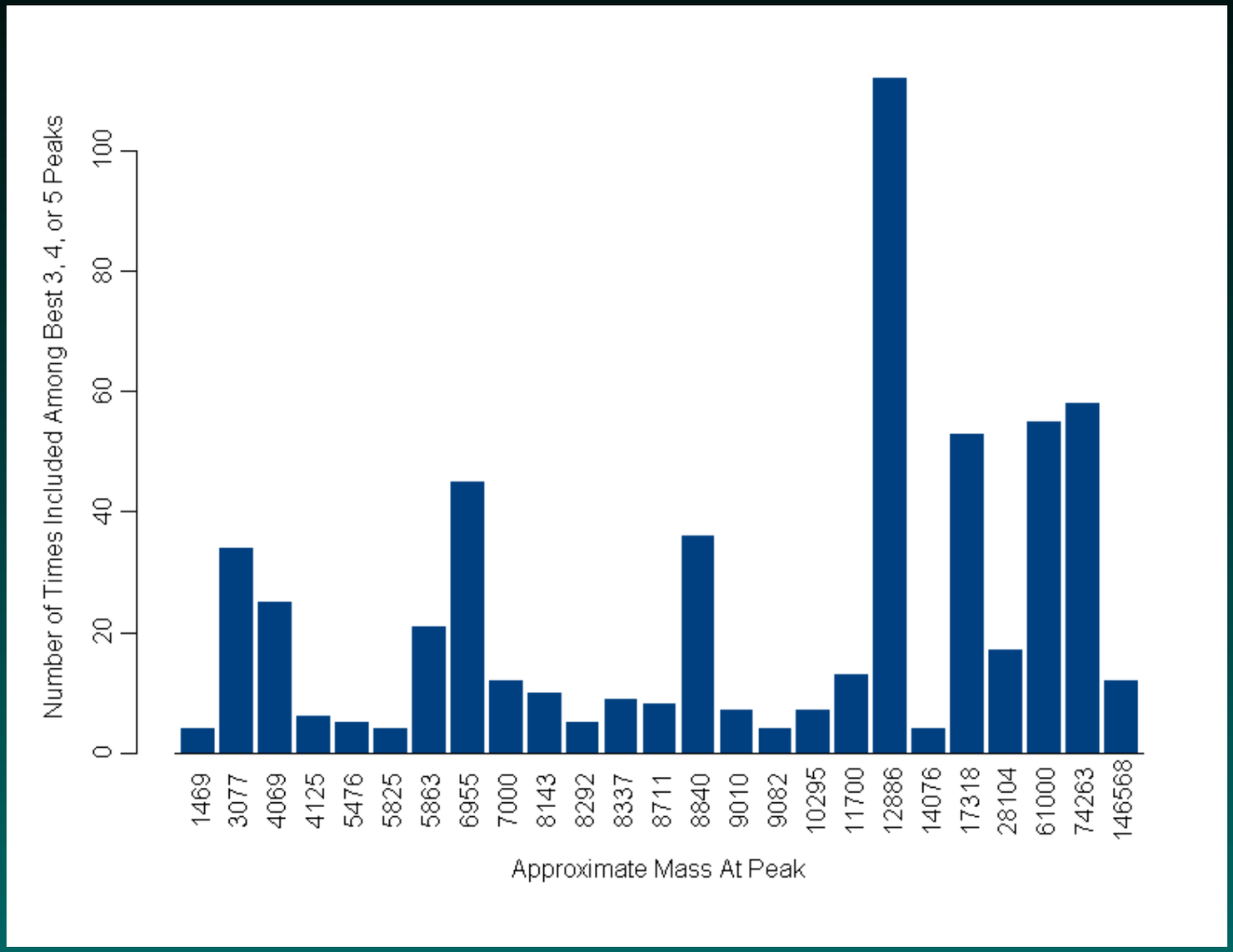
Find the Best Separators

Peaks	MD	P-Value	Wrong	LOOCV
12886	2.547	≤ 0.005	11	11
8840, 12886	5.679	≤ 0.01	5	6
3077, 12886 74263	9.019	≤ 0.01	3	4
5863, 8143 8840, 12886	12.585	≤ 0.01	3	3
4125, 7000 9010, 12886 74263	23.108	≤ 0.01	1	1

Remember the Randomness!

- GAs are a form of “directed random search”.
- Because of randomness in the algorithm, we can get different answers every time (and we do).
- Because there is no unique solution, we need to verify that the values we find are worth paying attention to.

How Often Did We Find the Best Peaks?



Survey the Results

There are 9 values that recur frequently, at masses of 3077, 4069, 5825, 6955, 8840, 12886, 17318, 61000, and 74263.

P-values are not from table lookups!

Simulating Significance

We weren't sure when a Mahalanobis distance would be "big" here, so we repeated the procedure with randomly generated data – noise matrices of size 506 by 41, and recomputing the MD values after evolving.

This is a slightly different use of simulation; earlier we used simulation to assess whether we were using cross-validation correctly, and here we're using it to assess whether the values we got were big.

The Challenge

We've specified a dataset, some methods of classification, and means of assessing classification accuracy (cross-validation).

Download the dataset yourself.

Find the best genes for comparing BRCA2+ cases with the others using 2 sample t-tests. Confirm that there are indeed 49 with p-values < 0.001.

Compute leave-one out cross-validation (LOOCV) classification accuracies for LDA, QDA, KNN ($k = 1, 3$), and DLDA.

Which is best?