Using pdmclass

James W. MacDonald

October 13, 2014

1 Overview

Classification is a statistical technique that uses measurements on a defined set of samples (a training set) to build a rule that can be used to infer the group membership of future samples. An example would be using gene expression data to classify cancer patients according to the expected response to a certain course of therapy.

There are many ways to build classification rules, but the general idea is the same; find patterns in the training set that are unique to each sample type and use this information to determine the class of new samples.

Microarrays hold great promise for building classifiers because of the amount of information that can be generated from each array. However, this is a two edged sword – much of the information cannot be distinguished from noise (low expressing genes), and having many more observations than samples can make the analysis computationally and statistically difficult. Therefore, it is usually desirable to pre-filter the genes down to a much smaller (100 - 200) set of genes before building the classifier. This itself is not a simple proposition – ideally this list should contain genes that are as uncorrelated as possible, because data from correlated genes is in some sense redundant information.

An alternative to manually subsetting the data is to use regularized regression models (partial least squares, ridge regression, principal components regression), which are designed to work with large numbers of correlated predictor variables. Since these methods are generally used in situations where the response is continuous (and in classification the response is categorical), we can use the optimal scoring algorithm of Hastie et al. (1994), which extends these methods to classification problems. For a more detailed description of these methods, please refer to Ghosh (2003).

2 A Simple Example

In this example we will use the *fibroEset* package, which contains expression data from Affymetrix HG-U95Av2 chips that were used to analyze early passage primary fibroblast cell lines from 18 human, 10 bonobo, and 11 gorilla samples. Although the utility of a classifier based on this data set is questionable at best, it does provide a workable example.

We first load the package:

```
> library("pdmclass")
> data("fibroEset")
> fibroEset

ExpressionSet (storageMode: lockedEnvironment)
assayData: 12625 features, 46 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: 1 2 ... 46 (46 total)
  varLabels: samp species
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
  pubMedIds: 12840040
Annotation: hgu95av2

> pData(fibroEset)

   samp species
1     1       b
2     2       b
3     3       b
4     4       b
5     5       b
6     6       b
7     7       b
8     8       b
9     9       b
10   10       b
11   11       b
12   12       g
```

```
13    13        g
14    14        g
15    15        g
16    16        g
17    17        g
18    18        g
19    19        g
20    20        g
21    21        g
22    22        g
23    23        g
24    24        h
25    25        h
26    26        h
27    27        h
28    28        h
29    29        h
30    30        h
31    31        h
32    32        h
33    33        h
34    34        h
35    35        h
36    36        h
37    37        h
38    38        h
39    39        h
40    40        h
41    41        h
42    42        h
43    43        h
44    44        h
45    45        h
46    46        h
```

First we fit a classifier to this data, using the expression values and the second column of the phenoData slot. Note here that in classical statistical applications the convention is for rows to contain subjects and columns to contain observations – we therefore have to transpose the expression data to meet this convention. In addition, we use the usual R formula interface

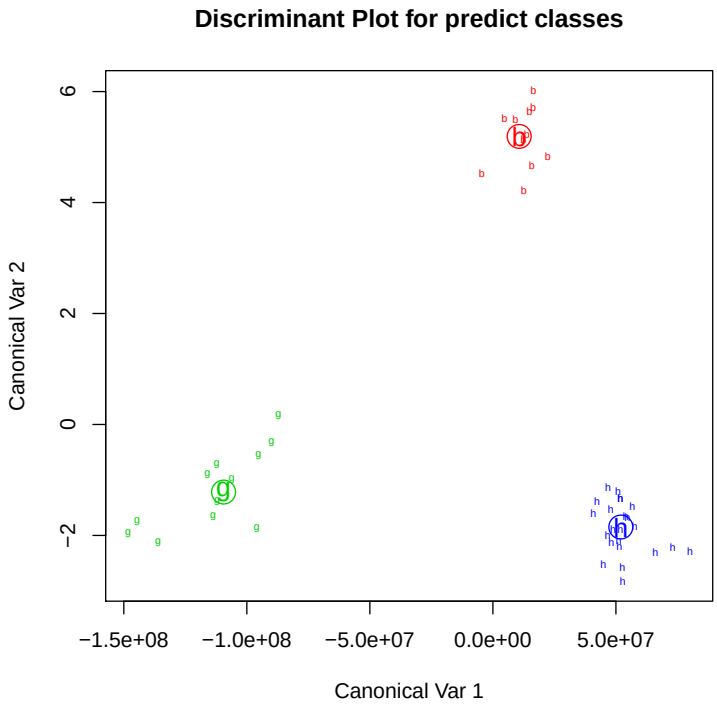

Figure 1: Plot of Fitted PLS Classifier

– for more information, see the `formula` help page.

```
> y <- as.factor(pData(fibroEset)[,2])
> x <- t(exprs(fibroEset))
> gn.class <- pdmClass(y ~ x, method = "pls")
```

Once we have fit the classifier we can make a plot that shows how well the samples are grouping.

```
> plot(gn.class, pch = levels(y))
```

Figure 1 shows the fitted pls classifier. As would be expected, the different species are quite well separated and very tightly grouped. Samples with more subtle differences would not be expected to group this nicely.

Having built the classifier, we will most likely want to use it to predict the class of new samples for which we don't know the classes *a priori*.

However, before we do this, it is prudent to test the classifier to see how accurate it is. We could simply take the data we used to build the classifier as if it were new data and predict the class of each sample.

```
> predict(gn.class)

 [1] b b b b b b b b b b b g g g g g g g g g g g g h h h h h h h h h h h h h h h h
[39] h h h h h h h
Levels: b g h
```

Since we are predicting the class of the data that was used to build the classifier, we expect that these results will be much better than what could be expected with a set of new data. To get an unbiased estimate of the accuracy, we need a 'test set'.

The canonical method of creating and testing a classifier is to split a set of data into a training and testing set. The training set is used to make the classifier and then the testing set is used to estimate the accuracy of the classifier by comparing the predicted class for each sample to the actual class. Unfortunately, it is often difficult to get sufficient numbers of samples to build an accurate classifier, so it may not be possible to split into a training and testing set. An alternative is to perform a 'leave one out' cross-validation where we remove a single sample and then build a classifier with the remaining samples. We then predict the class of the sample that was removed, repeating the process for each sample in turn. We will then have a vector of class assignments for each sample that we can compare to the true class membership to create a 'confusion matrix'.

```
> tst <- pdmClass.cv(y, x, method = "pls")
> confusion(tst, y)

         true
predicted  b  g  h
        b 11  0  5
        g  0 12  0
        h  0  0 18
attr(,"error")
[1] 0.1086957
```

Here we can see that we expect an error rate of approximately 0.109 when we apply this classifier to new samples.

After building a classifier, we may be interested in the genes that have the most influence in differentiating between sample types. For this we can use the `pdmGenes` function.

```
> gns <- featureNames(fibroEset)
> len <- 10
> tmp <- pdmGenes(y~x, genelist = gns, list.length = len, B=10)
> tmp

$`g vs b`
           X[[1L]]
33117_r_at     1.0
33659_at       1.0
31956_f_at     0.9
31505_at       1.0
32315_at       1.0
37307_at       1.0
36790_at       0.4
31509_at       0.5
36589_at       0.6
1598_g_at      0.3

$`h vs b`
           X[[2L]]
36666_at       1.0
34160_at       1.0
1385_at        1.0
39758_f_at     1.0
35905_s_at     1.0
31720_s_at     0.7
36790_at       0.8
41745_at       0.3
37185_at       0.8
38356_at       0.6
```

The `pdmGenes` function selects the top n genes (set by the argument *list.length*) from the original classifier, and then determines the importance of these genes by repeatedly taking bootstrap samples from the data and seeing what proportion of the time these genes are actually found to be influential in fitting a classifier using the bootstrap samples. The basic

idea being that a truly influential gene will repeatedly show up as being influential as we make slight perturbations to the data. Note that we are using *contr.treatment* contrasts for these model fits, so one set of samples will always be set as the baseline, and the output will list the genes influential in the comparison of the other samples to the baseline sample.

References

Debashis Ghosh. Penalized discriminant methods for the classification of tumors from gene expression data. *Biometrics*, 59:992–1000, 2003.

T. Hastie, R. Tibshirani, and A. Buja. Flexible discriminant analysis by optimal scoring. *Journal of the American Statistical Association*, 89:1255–1270, 1994.