Statistical Methods in Microarray Analysis Tutorial: Clustering and Discrimination

Jean Yee Hwa Yang

January 4, 2004

The purpose of this lab is two fold. The first is to perform some basic clustering and examine the effect of different cluster algorithms. The second aim is to build different classification procedures for tumor samples using gene expression data and to assess the accuracy of these classification procedures. We will use the dataset presented in Golub *et al.* (1999) and available at http://www.genome.wi.mit.edu/MPR. These data come from a study of gene expression in three types of acute leukemias: B-cell acute lymphoblastic leukemia (B-ALL), T-cell acute lymphoblastic leukemia (T-ALL) and acute myeloid leukemia (AML). Gene expression levels were measured for 38 B-cell ALL, 9 T-cell ALL and 25 AML tumor samples, using Affymetrix high-density oligonucleotide arrays hgu68a containing $p = 6817$ human genes.

You will work with data that have been pre-processed using procedures similar to the one described in Golub *et al.* (1999) and Dudoit *et al.* (2000). These steps are:

1. thresholding: floor of 100 and ceiling of 16,000;
2. filtering: exclusion of genes with $max/min \leq 5$ or $(max - min) \leq 500$ where max and min refer respectively to the maximum and minimum intensities for a particular gene across the mRNA samples;
3. base 2 logarithmic transformation (Golub *et al.* uses log 10 logarithmic transformation).

The **R** data file `GolubData.RData` contains gene expression levels and gene names. The filtered gene expression levels are stored in a 3571×72 matrix named `golub` with rows corresponding to genes and columns to mRNA samples.

You could proceed with this lab in two ways. For users who are familiar with **R** or **S-plus** language, you may prefer to work through this lab by writing your own code. For other users less familiar with **R**, you can use the `vExplorer` function from the `tkWidgets` package to step through this lab interactively. The `vExplorer` function provides a graphical interface for viewing and executing code chunks from the lab. To begin, start R and load the tutorial by typing the command:

```
> library(IMSLAB)
> vExplorer()
```

Next, select the `IMSLAB` package using the widget. A basic overview of **R** and Bioconductor WWW resources are provided at the end of this tutorial. In addition, you can find other Bioconductor tutorials and labs on this data set at `www.bioconductor.org`.

1 Getting started

Before we start the lab, here are some useful commands for getting help and sample scripts demonstrating software functionality

```
> help.start()
> apropos("mean")
> ? mean
> example("mean")
```

To load the data packages

```
> library(IMSLAB)
> data(GolubData)
```

2 Clustering

Cluster the leukemia mRNA samples using the `hclust` function and vary the number of genes. Do the T-ALL, B-ALL and AML samples cluster together? Try different between-clusters dissimilarity measures by modifying the "method" argument in `hclust`. Also try different dissimilarities metric by modifying the "method" argument in `dist` or come up with your own. Below are some questions to help you get started.

Q1. Perform a hierarchical clustering on the mRNA samples using correlation as similarity function and complete linkage agglomeration.

```
> library(mva)
> clust.cor <- hclust(as.dist(1 - cor(golub)), method = "complete")
> plot(clust.cor, cex = 0.6)
```

Q2. Perform a hierarchical clustering on the mRNA samples se Euclidean distance and average linkage agglomeration.

```
> clust.euclid <- hclust(dist(t(golub)), method = "average")
> plot(clust.euclid, cex = 0.6)
```

Q3. You might also like to create a cluster image using the function `heatmap` in the package `mva`. Keeping in mind that, this function performs hirarchical clustering on both geness and samples, this method will slow down considerably if the number of genes are too large. For illustration purposes, we have selected 100 variable genes.

```
> library(sma)
> golubvar <- apply(golub, 1, var, na.rm = TRUE)
> top100 <- stat.gnames(golubvar, 1:length(golubvar), crit = 100)$gnames
> heatmap(golub[top100, ])
```

Next, we can try different partition methods.
Q4. Perform kmeans clustering on the mRNA samples using correlation as the similarity function.

```
> clust.kmeans <- kmeans(as.dist(1 - cor(golub)), 3)
> names(clust.kmeans$cluster) <- colnames(golub)
> clust.kmeans$cluster[1:10]
```

```
 B-ALL:1  T-ALL:2  T-ALL:3  B-ALL:4  B-ALL:5  T-ALL:6  B-ALL:7
       1        1        1        1        2        1        1
 B-ALL:8  T-ALL:9 T-ALL:10
       1        1        1
```

Q5. Perform "Partition Around Medoids" clustering using the function `PAM` in the `cluster` library.

```
> library(cluster)
> clust.pam <- pam(as.dist(1 - cor(golub)), 3, diss = TRUE)
> clusplot(clust.pam, labels = 3, col.p = clust.pam$clustering)
```

Q6. Optional, use "Self Organizing Maps" (SOM) methods from the `som` library using a 2 by 2 grid size.

Q7. Optional, finally, we can also select the top 100 genes based on variance and perform the various clustering methods described above. Notice that we did not use any information associated with the samples during the gene selection process here. The function `stat.gnames` sorts genes according to the value of a statistic and in this case the statistics of interest is "variance".

```
> golubSub <- golub[top100, ]
> par(mfrow = c(2, 2))
> plot(hclust(as.dist(1 - cor(golubSub)), method = "complete"),
+     cex = 0.6)
> plot(hclust(dist(t(golubSub)), method = "average"), cex = 0.6)
> clust.pam <- pam(as.dist(1 - cor(golubSub)), 3, diss = TRUE)
> clusplot(clust.pam, labels = 3, col.p = clust.pam$clustering)
> par(mfrow = c(1, 1))
```

3 Discrimination

In the second part of this lab, we are interested in building classification rules to discriminate between the three tumor groups, B-ALL, T-ALL and AML. For ease of processing, the `golub` matrix was further separated into expression data for the learning and test set in matrix objects named `LS` and `TS` respectively. Different from the `golub` matrix, the matrix `LS` and `TS` are 38×3517 and 34×3517 respectively, with rows corresponding to tumor samples and columns corresponding to genes. The tumor class labels for the learning and test sets are given in vector objects named `LS.resp` and `TS.resp` respectively.

3.1 KNN

Q8. Lets begin by examining the **K Nearest neighbor classifer**. We will use the function `knn` from the `class` package); use `?knn` to see what options are available for this function. We want to see what happens with the different values of k. Using a simple feature selection procedure, we have selected the top 100 genes based on the between to within group sum of squares (BWSS) using the function `stat.bwss` in `sma` package.

```
> library(class)
> LS.bw <- stat.bwss(t(LS), as.integer(LS.resp))$bw
> LStop100 <- stat.gnames(LS.bw, 1:length(LS.bw), crit = 100)$gnames
> for (k in 4:7) {
+     disc.knn <- as.vector(knn(LS[, LStop100], TS[, LStop100],
+         cl = LS.resp, k))
+     print(paste("K = ", k))
+     print(table(disc.knn, TS.resp))
+ }

[1] "K =  4"
        TS.resp
disc.knn AML B-ALL T-ALL
   AML   14   0     1
   B-ALL  0  19     0
[1] "K =  5"
        TS.resp
disc.knn AML B-ALL T-ALL
   AML   14   0     0
   B-ALL  0  19     0
   T-ALL  0   0     1
[1] "K =  6"
        TS.resp
disc.knn AML B-ALL T-ALL
```

```
   AML   14   0     1
   B-ALL  0  19     0
[1] "K =  7"
        TS.resp
disc.knn AML B-ALL T-ALL
   AML   14   0     1
   B-ALL  0  19     0
```

Q9. Optional, if you are interested in producing similar pictures as shown in the lecture, you can try the functions `geneTS` and `plot.class2` provided with this lab. At the moment, this works only for bivariate data. A simple way is to choose two genes from the vector `LStop100`, alternatively, one can use the first and second principal components.

```
> newLS <- LS[, LStop100[2:3]]
> newTS <- geneTS(newLS)
> res.knn <- knn(newLS, newTS, cl = as.numeric(LS.resp),
+     k = 3)
> plot.class2(newLS, newTS, res.knn, cl = LS.resp)

NULL
```

3.2 Maximum likelihood discriminant rules

Q10. Use `stat.diag.da` from `sma` packages to build DLDA and DQDA classification procedures for tumor samples.

```
> disc.dqda <- stat.diag.da(LS[, LStop100], TS[, LStop100],
+     cl = as.integer(LS.resp), pool = 0)[[1]]
> table(disc.dqda, TS.resp)

         TS.resp
disc.dqda AML B-ALL T-ALL
        1 12   0     0
        2  2  19     0
        3  0   0     1

> disc.dlda <- stat.diag.da(LS[, LStop100], TS[, LStop100],
+     cl = as.integer(LS.resp), pool = 1)[[1]]
> table(disc.dlda, TS.resp)

         TS.resp
disc.dlda AML B-ALL T-ALL
        1 14   0     0
        2  0  19     0
        3  0   0     1
```

3.3 Trees

Q11. Use function `rpart` from `rpart` packages to build a classification tree. Compare the results from different classification rules.

```
> library(rpart)
> disc.rpart <- dorpart(LS = LS[, LStop100], TS = TS[,
+     LStop100], cl = as.numeric(LS.resp))
> table(disc.rpart, TS.resp)
```

```
          TS.resp
disc.rpart AML B-ALL T-ALL
         1 11   1     0
         2  3  18     0
         3  0   0     1
```

3.4 Error rate

Let us reconsider the K-nearest neighbor classifer.

Q12. Calculate the resubstitution error rate for a nearest neighbor using all the genes and k equals to 3.

```
> pred.LS <- as.vector(knn(LS, LS, cl = LS.resp, k = 3))
> table(pred.LS, LS.resp)
```

```
       LS.resp
pred.LS AML B-ALL T-ALL
  AML   11   0     0
  B-ALL  0  19     0
  T-ALL  0   0     8
```

```
> sum(pred.LS != LS.resp)
```

```
[1] 0
```

Q13. Calculate the test set error rate for a nearest neighbor using all the genes and k equals to 3.

```
> pred.TS <- as.vector(knn(LS, TS, cl = LS.resp, k = 3))
> table(pred.TS, TS.resp)
```

```
       TS.resp
pred.TS AML B-ALL T-ALL
  AML   13   0     0
  B-ALL  1  19     0
  T-ALL  0   0     1
```

```
> sum(pred.TS != TS.resp)

[1] 1
```

Q14. Lets consider another nearest neighbour classification rule using top 5 BWSS genes and k equals to 2. Calculate and compare the resubstitution and testset error. Notice how in this case, the test set error is much higher than the resubsitution error.

```
> LStop5 <- stat.gnames(LS.bw, 1:length(LS.bw), crit = 5)$gnames
> pred.LS <- as.vector(knn(LS[, LStop5], LS[, LStop5],
+     cl = LS.resp, k = 2))
> table(pred.LS, LS.resp)

       LS.resp
pred.LS AML B-ALL T-ALL
  AML   11   0     0
  B-ALL  0  19     1
  T-ALL  0   0     7

> sum(pred.LS != LS.resp)

[1] 1

> pred.TS <- as.vector(knn(LS[, LStop5], TS[, LStop5],
+     cl = LS.resp, k = 2))
> table(pred.TS, TS.resp)

       TS.resp
pred.TS AML B-ALL T-ALL
  AML   13   2     0
  B-ALL  1  16     0
  T-ALL  0   1     1

> sum(pred.TS != TS.resp)

[1] 4
```

Q15. Optional. Build a nearest neighbor classifier with k selected by cross-validation? Here is an example function build on the function `knn.cv`.

```
> knn.cvk <- function(LS, cl, k = 1:5, ...) {
+     cv.err <- cl.pred <- c()
+     for (i in k) {
+         newpre <- as.vector(knn.cv(LS, cl, i))
+         cl.pred <- cbind(cl.pred, newpre)
```

```
+         cv.err <- c(cv.err, sum(cl != newpre))
+     }
+     k0 <- k[which.min(cv.err)]
+     return(list(k = k0, pred = cl.pred[, which.min(cv.err)]))
+ }
> knn.cvk(LS, cl = LS.resp, k = 3:7)
$k
[1] 4

$pred
 [1] "B-ALL" "T-ALL" "T-ALL" "B-ALL" "B-ALL" "T-ALL" "B-ALL" "B-ALL"
 [9] "T-ALL" "T-ALL" "T-ALL" "AML"   "B-ALL" "T-ALL" "B-ALL" "B-ALL"
[17] "B-ALL" "B-ALL" "B-ALL" "B-ALL" "B-ALL" "B-ALL" "T-ALL" "B-ALL"
[25] "B-ALL" "B-ALL" "B-ALL" "AML"   "AML"   "AML"   "AML"   "AML"
[33] "AML"   "AML"   "AML"   "AML"   "AML"   "AML"
```

Finally, build a nearest neighbour classifer with the number of features selected by cross-validation. You can check out the function `knn.var` from the `knnTree` package.

4 Annotation

Q16. Optional. Using the Affymetrix or EBI web site to find out the gene names associated with the Affymetrix IDs

`X04145_at and M27891_at.`

5 R and Bioconductor WWW resources

For software and documentation consult

- Main R project website: `www.R-project.org`.
- Comprehensive R Archive Network (CRAN): `cran.r-project.org`.
  Base system and contributed packages (Linux, MacOS, Windows), manuals, tutorials, R News.
  For Windows, there is an installer for the main R system. Contributed packages from CRAN can be installed using the "Install package from CRAN ..." item on the "Packages" menu. For other packages, you can use the "Install package from local zip file ..." item.
- Bioconductor website: `www.bioconductor.org`.
  Software packages, vignettes, datasets, short course materials.
  To install Bioconductor packages for windows, use the "Install package(s) from Bioconductor ..." item on the "Packages" menu.

Acknowledgements

This lab was modified from a previous classification lab perpared by Sandrine Dudoit. Thanks also to Jane Fridlyand for discussion during the prepartion of this lab.

References

1. Golub *et. al.* (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* 286(5439): 531-537.

2. Dudoit, S., Fridlyand, J. and Speed, T. P. (2002). Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, Vol. 97, No. 457, p. 77-87.