Data Mining in R
Final Project

Kevin Zollicoffer

2/10/2014

Assignment

The framework provided in the DMwR packaged is leveraged throughtout the assigment analysis.

Source Code

Complete source code for this project can be found on github at
`https://github.com/zollie/PASS/tree/master/DMinR/MicroarrayClassification`

Preparation

First we wrap the knn() function in an interface that can be used more readily with the tools in the DMwR package.

```
> kNN <- function(form,train,test,norm=T,norm.stats=NULL,...) {
+   require(class,quietly=TRUE)
+   tgtCol <- which(colnames(train)==as.character(form[[2]]))
+   if (norm) {
+     if (is.null(norm.stats)) tmp <- scale(train[,-tgtCol],center=T,scale=T)
+     else tmp <- scale(train[,-tgtCol],center=norm.stats[[1]],scale=norm.stats[[2]])
+     train[,-tgtCol] <- tmp
+     ms <- attr(tmp,"scaled:center")
+     ss <- attr(tmp,"scaled:scale")
+     test[,-tgtCol] <- scale(test[,-tgtCol],center=ms,scale=ss)
+   }
+   knn(train[,-tgtCol],test[,-tgtCol],train[,tgtCol],...)
+ }
```

Because the tasks carried out by each learner are similar, we create a generic function to run each of them. This function will be called by the variants() function. variants() is in turn called by experimentalComparison(). The genericModel() function uses a helper function to calculate the Inter-Quartile Range of a row so we define that first.

```
> rowIQRs <- function(em)
+   rowQ(em,ceiling(0.75*ncol(em))) - rowQ(em,floor(0.25*ncol(em)))
> genericModel <- function(form,train,test,
+                          learner,
+                          fs.meth,
+                          ...)
+ {
+   cat('=')
+   tgt <- as.character(form[[2]])
+   tgtCol <- which(colnames(train)==tgt)
+
+   if (learner == 'knn')
+     pred <- kNN(form,
+                 train,
+                 test,
+                 norm.stats=list(rowMedians(t(as.matrix(train[,-tgtCol]))),
+                                 rowIQRs(t(as.matrix(train[,-tgtCol])))),
+                 ...)
+   else {
+     model <- do.call(learner,c(list(form,train),list(...)))
+     pred <- if (learner != 'randomForest') predict(model,test)
+     else predict(model,test,type='response')
+   }
+
+   #c(accuracy=ifelse(pred == resp(form,test),100,0))
+   structure(c(accuracy=ifelse(pred == resp(form,test),100,0)), itInfo=list(pred))
+ }
```

Run

Here we load the data we are going to use in the comparison experiment. For simplicity a random sample of 30 genes is used in the experimental comparison.

```
> library(Biobase)
> library(DMwR)
> library(class)
> library(randomForest)
> load('myALL.Rdata')
> set.seed(1234) # for reproducibility
> data <- exprs(ALLb)
> # random sample of 30 genes
> rows.train <- sample(nrow(data), 30)
> data.train <- data[rows.train,]
> dt <- data.frame(t(data.train),Mut=ALLb$mol.bio)
> # the formula
> DSs <- list(dataset(Mut ~ .,dt,'ALL'))
```

We are now ready to run the comparison experiment. Below we will use knn, and a random forest ensemble with 3 variants each. We will vary the randomForest ensemble by the number of trees grown $(250, 500, 1000)$, and predictors sampled for node split $(5, 15, 30)$. We will vary knn only by k $(3, 5, 7, 11)$.

```
> vars <- list()
> # model variants
> vars$randomForest <- list(ntree=c(250,500,1000),
+                           mtry=c(5,15,30))
> vars$knn <- list(k=c(3,5,7,11))
```

The experiment is run in the loop below by setting up each learner and calling variants from within experimentalComparison(). The resulting objects of class compExp are then stored on the file system for later evaluation.

```
> # The learners to evaluate
> TODO <- c('knn','randomForest')
> for(td in TODO) {
+   assign(td,
+          experimentalComparison(
+            DSs,
+            c(
+              do.call('variants',
+                      c(list('genericModel',learner=td),
+                        vars[[td]],
+                        varsRootName=td))
+            ),
+            loocvSettings(seed=1234,verbose=F),
+            'itsInfo'=T
+          )
+   )
+   save(list=td,file=paste(td,'Rdata',sep='.'))
+ }

#####  LOOCV  EXPERIMENTAL COMPARISON #####

** DATASET :: ALL

++ LEARNER :: genericModel  variant ->  knn.v1

 LOOCV experiment with verbose =  FALSE  and seed = 1234
==========================================================================================

++ LEARNER :: genericModel  variant ->  knn.v2

 LOOCV experiment with verbose =  FALSE  and seed = 1234
==========================================================================================
```

```
++ LEARNER :: genericModel  variant ->  knn.v3

 LOOCV experiment with verbose =  FALSE  and seed = 1234
================================================================================

++ LEARNER :: genericModel  variant ->  knn.v4

 LOOCV experiment with verbose =  FALSE  and seed = 1234
================================================================================

#####  LOOCV  EXPERIMENTAL COMPARISON #####

** DATASET :: ALL

++ LEARNER :: genericModel  variant ->  randomForest.v1

 LOOCV experiment with verbose =  FALSE  and seed = 1234
================================================================================

++ LEARNER :: genericModel  variant ->  randomForest.v2

 LOOCV experiment with verbose =  FALSE  and seed = 1234
================================================================================

++ LEARNER :: genericModel  variant ->  randomForest.v3

 LOOCV experiment with verbose =  FALSE  and seed = 1234
================================================================================

++ LEARNER :: genericModel  variant ->  randomForest.v4

 LOOCV experiment with verbose =  FALSE  and seed = 1234
================================================================================

++ LEARNER :: genericModel  variant ->  randomForest.v5

 LOOCV experiment with verbose =  FALSE  and seed = 1234
================================================================================

++ LEARNER :: genericModel  variant ->  randomForest.v6

 LOOCV experiment with verbose =  FALSE  and seed = 1234
================================================================================

++ LEARNER :: genericModel  variant ->  randomForest.v7
```

```
 LOOCV experiment with verbose =  FALSE  and seed = 1234
==========================================================================================

++ LEARNER :: genericModel  variant ->  randomForest.v8

 LOOCV experiment with verbose =  FALSE  and seed = 1234
==========================================================================================

++ LEARNER :: genericModel  variant ->  randomForest.v9

 LOOCV experiment with verbose =  FALSE  and seed = 1234
==========================================================================================
```

Evaluation

We use the tools provided in DMwR to evaluate model performance. First we load the compExp objects from the filesystem.

```
> load('knn.Rdata')
> load('randomForest.Rdata')
> all.trials <- join(knn,randomForest,by='variants')
```

A summary of the results can be obtained with

```
> summary(all.trials)

== Summary of a   Experiment ==

 LOOCV experiment with verbose =  FALSE  and seed = 1234

* Data sets ::  ALL
* Learners  ::  knn.v1, knn.v2, knn.v3, knn.v4, randomForest.v1, randomForest.v2, randomFore

* Summary of Experiment Results:


-> Datataset:  ALL

        *Learner: knn.v1
         accuracy
avg      42.55319
std      49.70745
min       0.00000
max     100.00000
invalid   0.00000
```

```
        *Learner: knn.v2
         accuracy
avg      40.42553
std      49.33787
min       0.00000
max     100.00000
invalid   0.00000

        *Learner: knn.v3
         accuracy
avg      42.55319
std      49.70745
min       0.00000
max     100.00000
invalid   0.00000

        *Learner: knn.v4
         accuracy
avg      39.36170
std      49.11712
min       0.00000
max     100.00000
invalid   0.00000

        *Learner: randomForest.v1
         accuracy
avg      41.48936
std      49.53455
min       0.00000
max     100.00000
invalid   0.00000

        *Learner: randomForest.v2
         accuracy
avg      40.42553
std      49.33787
min       0.00000
max     100.00000
invalid   0.00000

        *Learner: randomForest.v3
         accuracy
avg      39.36170
std      49.11712
min       0.00000
max     100.00000
```

```
invalid   0.00000

        *Learner: randomForest.v4
         accuracy
avg      44.68085
std      49.98284
min       0.00000
max     100.00000
invalid   0.00000

        *Learner: randomForest.v5
         accuracy
avg      41.48936
std      49.53455
min       0.00000
max     100.00000
invalid   0.00000

        *Learner: randomForest.v6
         accuracy
avg      41.48936
std      49.53455
min       0.00000
max     100.00000
invalid   0.00000

        *Learner: randomForest.v7
         accuracy
avg      43.61702
std      49.85681
min       0.00000
max     100.00000
invalid   0.00000

        *Learner: randomForest.v8
         accuracy
avg      44.68085
std      49.98284
min       0.00000
max     100.00000
invalid   0.00000

        *Learner: randomForest.v9
         accuracy
avg      43.61702
std      49.85681
```

```
min       0.00000
max     100.00000
invalid   0.00000
```

The top 10 variants were

```
> rankSystems(all.trials,top=10,max=T)

$ALL
$ALL$accuracy
            system    score
1  randomForest.v4 44.68085
2  randomForest.v8 44.68085
3  randomForest.v7 43.61702
4  randomForest.v9 43.61702
5           knn.v1 42.55319
6           knn.v3 42.55319
7  randomForest.v1 41.48936
8  randomForest.v5 41.48936
9  randomForest.v6 41.48936
10          knn.v2 40.42553
```

The best model was

```
> best <- getVariant('randomForest.v4', all.trials)
> best

Learner::  "genericModel"

Parameter values
         learner  =  "randomForest"
         ntree  =  250
         mtry  =  15
```

Interpretation

Clearly, reducing the the number of features to 30 randomly leads to inferior models than that produced in Chapter 5 where feature reduction is achieved via ANOVA filtering and random forest importance ranking. The best performing model in our random feature selection regime is randomForest.v4 with an accuracy score of 44.68. This compares with Chapter 5 scores where features were reduced using ANOVA filtering and random forest feature importance ranking of 88.29 for knn where $k = 5$.

Our variant model comparison test without ANOVA filtering and random forest feature selection did run considerably faster however.