Practical Data Science: Reducing High Dimensional Data in R

Manuel Amunategui - amunategui@gmail.com

Feature Selection using Ensembles and mRMRe: an R package for parallelized mRMR ensemble feature selection

The **Minimum Redundancy Maximum Relevance (mRMR)** package is based on Minimum redundancy feature selection (https://en.wikipedia.org/wiki/Minimum_redundancy_feature_selection) and is a very popular tool in bio-statistics and genetic research to sort through data sets contain thousands to millions of features. It is known to be very fast and to do a better job that the classical forward, backward, mixed selection approaches.

For more information on the actual R package see this vignette from its authors (https://cran.r-project.org/web/packages/mRMRe/vignettes/mRMRe.pdf).

I won't go into much detail here, instead, I'll just show you how to apply it to our **Gisette** data set (the same data set as in the earlier lectures Gisette (https://archive.ics.uci.edu/ml/datasets/Gisette)).

```
library(RCurl) # download https data

urlfile <- 'https://archive.ics.uci.edu/ml/machine-learning-databases/gisette/GISE
TTE/gisette_train.data'
x <- getURL(urlfile, ssl.verifypeer = FALSE)
gisetteRaw <- read.table(textConnection(x), sep = '', header= FALSE, stringsAsFact
ors = FALSE)

urlfile <- "https://archive.ics.uci.edu/ml/machine-learning-databases/gisette/GISE
TTE/gisette_train.labels"
x <- getURL(urlfile, ssl.verifypeer = FALSE)
g_labels <- read.table(textConnection(x), sep = '', header = FALSE, stringsAsFacto
rs = FALSE)
```

Now we create a full data set with the outcome variable (what we're trying to predict) as last feature. We also remove all duplicate columns from the set and fix the outcome to be between 0 and 1:

```
# build data set
gisette_df <- cbind(as.data.frame(sapply(gisetteRaw, as.numeric)), cluster=g_label
s$V1)

# remove duplicate columns from entire data set
# http://stackoverflow.com/questions/9818125/identifying-duplicate-columns-in-an-
r-data-frame
gisette_df <- gisette_df[!duplicated(lapply(gisette_df, summary))]

# turn outcome to classic format 0,1 instead of -1,1
gisette_df$cluster <- ifelse(gisette_df$cluster==-1,0,1)
```

We need to split the data into three sets, one for training, one for testing, and another for validating:

```
set.seed(1234)
split <- sample(nrow(gisette_df), floor(0.5*nrow(gisette_df)))
gisette_df_train_test <- gisette_df[split,]
gisette_df_validate <- gisette_df[-split,]

# split gisette_df_train_test data set into training and testing
set.seed(1234)
split <- sample(nrow(gisette_df_train_test), floor(0.5*nrow(gisette_df_train_tes
t)))
traindf <- gisette_df_train_test[split,]
testdf <- gisette_df_train_test[-split,]
```

We now call the **mRMRe** function on our data set. We ask for the top 20 features and present the material using scores and feature names:

```
# install the package if you don't already have
# install.packages("mRMRe")

library(mRMRe)
mRMR_data <- mRMR.data(data = traindf)

print(mRMR_data)
```

```
## Formal class 'mRMRe.Data' [package "mRMRe"] with 7 slots
##   ..@ sample_names : chr [1:1500] "1541" "624" "5414" "4480" ...
##   ..@ feature_names: chr [1:2808] "V1" "V2" "V3" "V4" ...
##   ..@ feature_types: num [1:2808] 0 0 0 0 0 0 0 0 0 0 ...
##   ..@ data         : num [1:1500, 1:2808] 0 0 805 0 0 0 0 836 0 0 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:1500] "1541" "624" "5414" "4480" ...
##   .. .. ..$ : chr [1:2808] "V1" "V2" "V3" "V4" ...
##   ..@ strata       : num [1:1500] 0 0 0 0 0 0 0 0 0 0 ...
##   ..@ weights      : num [1:1500] 1 1 1 1 1 1 1 1 1 1 ...
##   ..@ priors       : num[0 , 0 ]
```

```
# classic example
feats <- mRMR.classic(data = mRMR_data, target_indices = c(ncol(traindf)),
                      feature_count = 20)

bestVars <-data.frame('features'=names(traindf)[solutions(feats)[[1]]], 'scores'=
scores(feats)[[1]])
print(bestVars)
```

```
##    features  scores
## 1     V3657 0.30516
## 2      V949 0.03827
## 3     V3066 0.05191
## 4      V468 0.04160
## 5      V558 0.08141
## 6     V3828 0.05060
## 7     V3003 0.05192
## 8     V3976 0.05701
## 9     V4508 0.03639
## 10    V4446 0.03483
## 11    V1359 0.03208
## 12    V3063 0.04000
## 13    V4165 0.04041
## 14    V1229 0.03435
## 15     V215 0.03046
## 16     V512 0.03048
## 17    V3966 0.02607
## 18    V4354 0.02468
## 19     V456 0.02517
## 20    V3544 0.02816
```

So, how did we do with our 10 best? Here's something new, we'll try with both a GLMNET model (https://en.wikipedia.org/wiki/Generalized_linear_model) and a k-nearest neighbors algorithm (KNN) model (https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm) (a very easy switch using the caret package):

```
library(caret)

traindf_temp <- traindf[c(as.character(bestVars$features), 'cluster')]
# caret requires a factor of non-numeric value
traindf_temp$cluster <- ifelse(traindf_temp$cluster == 1, "yes", "no")
traindf_temp$cluster <- as.factor(traindf_temp$cluster )

objControl <- trainControl(method='cv', number=3, returnResamp='none', summaryFunc
tion = twoClassSummary, classProbs = TRUE)
glmnet_model <- train(cluster~., data=traindf_temp, method="glmnet",
                      metric='roc', trControl=objControl)
glmnet_predictions <- predict(object=glmnet_model, newdata= gisette_df_validate[,a
s.character(bestVars$features)], type='raw')

# caret requires a factor of non-numeric value
gisette_df_validate$cluster <- ifelse(gisette_df_validate$cluster == 1, "yes", "n
o")
gisette_df_validate$cluster <- as.factor(gisette_df_validate$cluster )
print(postResample(pred=glmnet_predictions, obs=gisette_df_validate$cluster))
```

```
## Accuracy    Kappa
##   0.9170   0.8340
```

```
knn_model <- train(cluster~., data=traindf_temp, method="knn",
                       metric='roc', trControl=objControl)

knn_predictions <- predict(object=knn_model, newdata= gisette_df_validate[,as.char
acter(bestVars$features)], type='raw')
print(postResample(pred=knn_predictions, obs=as.factor(gisette_df_validate$cluste
r)))
```

```
## Accuracy    Kappa
##   0.9306   0.8613
```

Here is another interesting feature of the **mRMRe** library, it can run an ensemble of smaller sampled sets of the data set and return the importance of each feature for each run.

```
# ensemble example
feats <- mRMR.ensemble(data = mRMR_data, target_indices = c(ncol(traindf)),
                         solution_count = 5,feature_count = 10)
bestVars <-data.frame('features'=names(traindf)[solutions(feats)[[1]]], 'scores'=
scores(feats)[[1]])
print(bestVars)
```

```
##    features scores.1 scores.2 scores.3 scores.4 scores.5
## 1      V512  0.19235  0.20771  0.24649  0.26516  0.30516
## 2      V949  0.07883  0.10540  0.12611  0.08010  0.03827
## 3      V905  0.06110  0.11331  0.06697  0.07170  0.05191
## 4     V3976  0.08167  0.04468  0.04047  0.05288  0.04160
## 5     V4165  0.06339  0.04406  0.09207  0.04333  0.08141
## 6     V4446  0.04076  0.08954  0.04353  0.06707  0.05060
## 7      V558  0.06802  0.05126  0.03277  0.04846  0.05192
## 8     V4508  0.03776  0.03729  0.05405  0.04343  0.05701
## 9      V468  0.03565  0.03383  0.05119  0.03362  0.03639
## 10    V3063  0.04546  0.04696  0.04128  0.03721  0.03483
## 11    V4508  0.19235  0.20771  0.24649  0.26516  0.30516
## 12    V4387  0.07883  0.10540  0.12611  0.08010  0.03827
## 13    V3003  0.06110  0.11331  0.06697  0.07170  0.05191
## 14    V3966  0.08167  0.04468  0.04047  0.05288  0.04160
## 15     V468  0.06339  0.04406  0.09207  0.04333  0.08141
## 16    V3657  0.04076  0.08954  0.04353  0.06707  0.05060
## 17     V720  0.06802  0.05126  0.03277  0.04846  0.05192
## 18     V558  0.03776  0.03729  0.05405  0.04343  0.05701
## 19    V4446  0.03565  0.03383  0.05119  0.03362  0.03639
## 20    V4165  0.04546  0.04696  0.04128  0.03721  0.03483
## 21    V3976  0.19235  0.20771  0.24649  0.26516  0.30516
## 22    V3003  0.07883  0.10540  0.12611  0.08010  0.03827
## 23    V4165  0.06110  0.11331  0.06697  0.07170  0.05191
## 24     V468  0.08167  0.04468  0.04047  0.05288  0.04160
## 25    V3657  0.06339  0.04406  0.09207  0.04333  0.08141
## 26     V949  0.04076  0.08954  0.04353  0.06707  0.05060
## 27    V2100  0.06802  0.05126  0.03277  0.04846  0.05192
## 28     V558  0.03776  0.03729  0.05405  0.04343  0.05701
## 29    V1229  0.03565  0.03383  0.05119  0.03362  0.03639
## 30     V456  0.04546  0.04696  0.04128  0.03721  0.03483
## 31     V558  0.19235  0.20771  0.24649  0.26516  0.30516
## 32    V3066  0.07883  0.10540  0.12611  0.08010  0.03827
## 33    V4508  0.06110  0.11331  0.06697  0.07170  0.05191
## 34     V949  0.08167  0.04468  0.04047  0.05288  0.04160
## 35     V468  0.06339  0.04406  0.09207  0.04333  0.08141
## 36    V3657  0.04076  0.08954  0.04353  0.06707  0.05060
## 37    V3003  0.06802  0.05126  0.03277  0.04846  0.05192
## 38    V3976  0.03776  0.03729  0.05405  0.04343  0.05701
## 39    V4446  0.03565  0.03383  0.05119  0.03362  0.03639
## 40     V456  0.04546  0.04696  0.04128  0.03721  0.03483
## 41    V3657  0.19235  0.20771  0.24649  0.26516  0.30516
## 42     V949  0.07883  0.10540  0.12611  0.08010  0.03827
## 43    V3066  0.06110  0.11331  0.06697  0.07170  0.05191
## 44     V468  0.08167  0.04468  0.04047  0.05288  0.04160
## 45     V558  0.06339  0.04406  0.09207  0.04333  0.08141
## 46    V3828  0.04076  0.08954  0.04353  0.06707  0.05060
## 47    V3003  0.06802  0.05126  0.03277  0.04846  0.05192
## 48    V3976  0.03776  0.03729  0.05405  0.04343  0.05701
## 49    V4508  0.03565  0.03383  0.05119  0.03362  0.03639
```

```
## 50    V4446  0.04546  0.04696  0.04128  0.03721  0.03483
```