Practical Data Science: Reducing High Dimensional Data in R

Manuel Amunategui - amunategui@gmail.com

Feature Selection - Variable Selection with GBM

So, what if you want to reduce you data's high dimensionality but still need to preserve you variables?

PCA variables won't do the trick as they're synthetic and made up of bits and pieces of other variables. A common solution to this conundrum is variable selection (https://en.wikipedia.org/wiki/Feature_selection). This is the process of taking a sample of your wide data set and attempt to find the most valuable ones.

We'll keep working with the wonderful library caret (http://topepo.github.io/caret/index.html) and use its great function **varImp**.

We're going to use two models: gbm (Generalized Boosted Models) (https://en.wikipedia.org/wiki/Gradient_boosting) and glmnet (Generalized Linear Models) (https://en.wikipedia.org/wiki/Generalized_linear_model). Approaching a new data set using different models is one way of getting a handle on your data. **gbm uses boosted trees while glmnet uses regression. (Note: gbm can handle NAs but glmnet cannot).**

We're also going to keep working with 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/GISETTE/gisette_train.data'
x <- getURL(urlfile, ssl.verifypeer = FALSE)
gisetteRaw <- read.table(textConnection(x), sep = '', header= FALSE, stringsAsFactors = FALSE)

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

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

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,]
```

GBM

Let's start with a GBM model. We will use the trainControl (http://www.inside-r.org/packages/cran/caret/docs/trainControl) function from caret. This function will cross-validate our data in order to find the best parameters for the GBM model. Warning: this takes a while…

```
# caret requires a factor of non-numeric value
traindf$cluster <- ifelse(traindf$cluster == 1, "yes", "no")
traindf$cluster <- as.factor(traindf$cluster )

library(caret)
fitControl <- trainControl(method='cv', number=3, returnResamp='none', verboseIter
= FALSE,
                           summaryFunction = twoClassSummary, classProbs = TRUE)
gbm_model <- train(cluster~., data=traindf, trControl=fitControl, method="gbm",
                   metric='roc')
```

```
print(gbm_model)
```

```
## Stochastic Gradient Boosting
##
## 1500 samples
## 5000 predictors
##    2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
##
## Summary of sample sizes: 1001, 999, 1000
##
## Resampling results across tuning parameters:
##
##  interaction.depth  n.trees  ROC        Sens       Spec       ROC SD       Sens
SD       Spec SD
##  1                   50      0.9624747  0.8694918  0.9352547  0.007416151  0.01
0626321  0.019833990
##  1                  100      0.9788316  0.9107064  0.9417247  0.007507123  0.00
6458501  0.020140583
##  1                  150      0.9839339  0.9381809  0.9455906  0.004461157  0.00
4251992  0.021682348
##  2                   50      0.9835423  0.9189538  0.9430016  0.005796000  0.01
0380782  0.008134521
##  2                  100      0.9877839  0.9478114  0.9507737  0.002950088  0.00
6184166  0.011907544
##  2                  150      0.9890942  0.9505549  0.9533627  0.001899431  0.00
7074401  0.014052696
##  3                   50      0.9877592  0.9519210  0.9533677  0.005299019  0.01
0370601  0.011673619
##  3                  100      0.9898549  0.9574306  0.9559567  0.001869422  0.01
0293466  0.009803711
##  3                  150      0.9910399  0.9560589  0.9585508  0.002614481  0.01
2512957  0.009781739
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## ROC was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 150, interaction.depth = 3,
shrinkage = 0.1 and n.minobsinnode = 10.
```

By print the gbm_model object, we get the best settings at the bottom of the report:

Tuning parameter ‘shrinkage’ was held constant at a value of 0.1 ROC was used to select the optimal model using the largest value. The final values used for the model were n.trees = 150, interaction.depth = 3 and shrinkage = 0.1.

```
testdf$cluster <- ifelse(testdf$cluster == 1, "yes", "no")
testdf$cluster <- as.factor(testdf$cluster )
predictions <- predict(object=gbm_model, testdf[,setdiff(names(testdf), 'cluste
r')], type='raw')
head(predictions)
```

```
## [1] yes yes no  yes yes no
## Levels: no yes
```

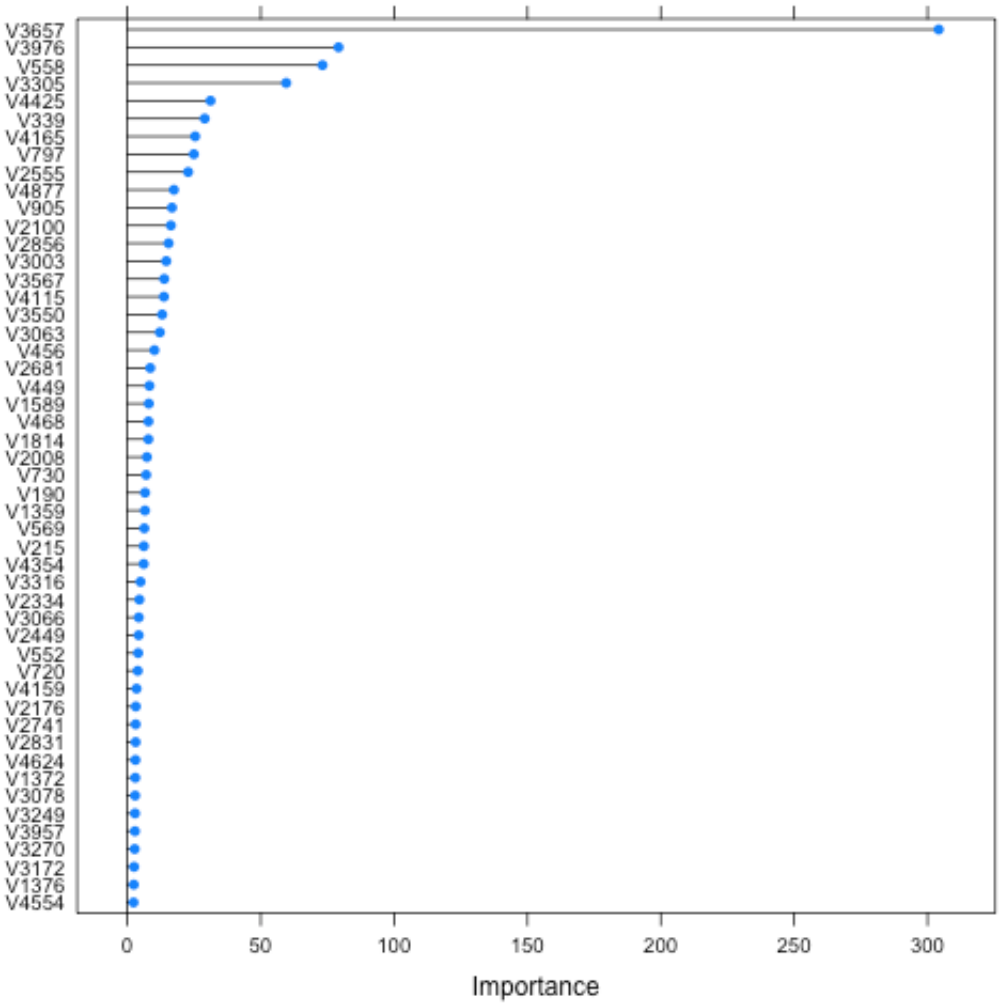
```
print(postResample(pred=predictions, obs=testdf$cluster))
```

```
## Accuracy    Kappa
##   0.9573   0.9146
```

We get 95.73%. Now, let's see how many of these variables we can drop and still get a great accuracy score. If you print the summary of gbm_model, you will get a list and graph of the best variables. Unfortunately, this isn't convenient here as there are 5000 variables.

This is where the varImp (http://topepo.github.io/caret/varimp.html) function comes in handy. Let's plot only the top 50 best variables:

```
# only plot top 20 variables
plot(varImp(gbm_model,scale=F), top = 50)
```



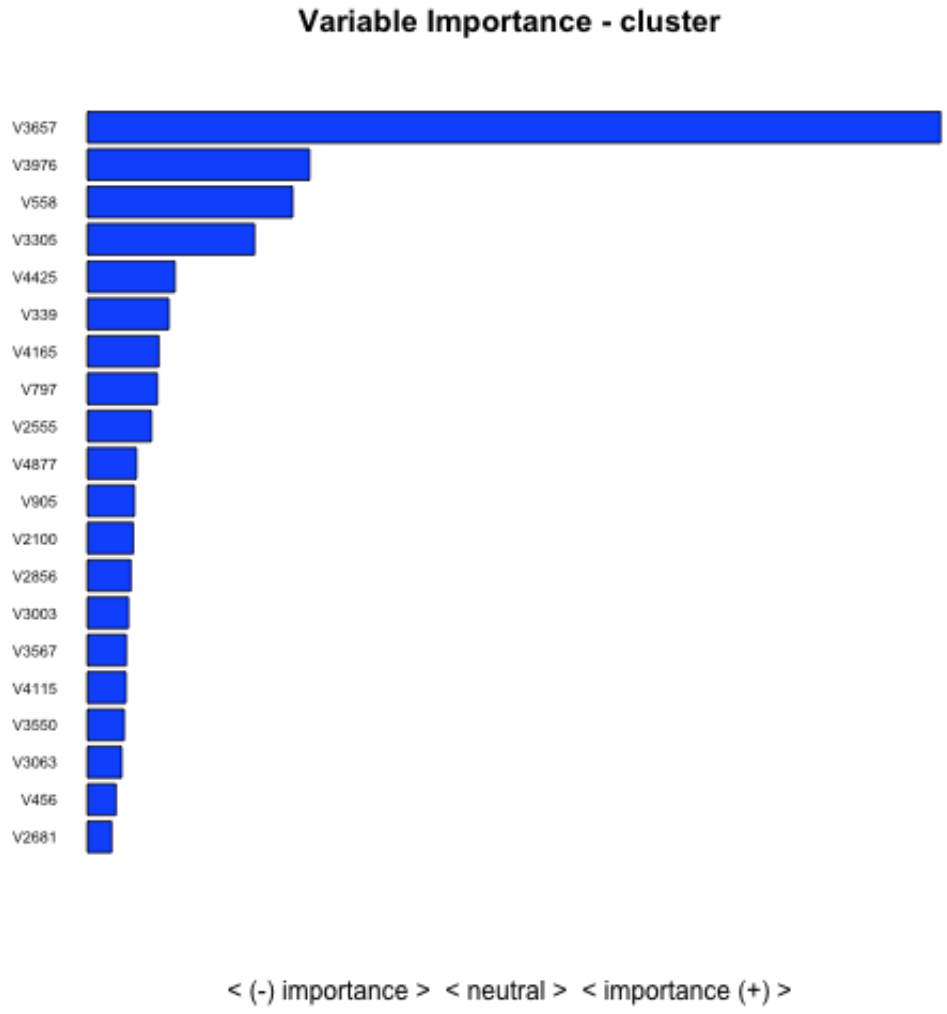
Here you see that V3657 is the most predictive feature for the outcome 'cluster'. No, with a little handy work, we can do a much better job at presenting the information:

```
# display variable importance on a +/- scale

vimp <- varImp(gbm_model, scale=F)
results <- data.frame(row.names(vimp$importance),vimp$importance$Overall)
results$VariableName <- rownames(vimp)
colnames(results) <- c('VariableName','Weight')
results <- results[order(results$Weight),]
# we do not want factors, just characters
results$VariableName <- as.character(results$VariableName)

# let's display the best 20 features
results_temp <- tail(results,20)
par(mar=c(5,5,4,2)) # increase y-axis margin.

xx <- barplot(results_temp$Weight, width = 0.85,
              main = paste("Variable Importance -",'cluster'), horiz = T,
              xlab = "< (-) importance >  < neutral >  < importance (+) >", axes =
FALSE,
              col = ifelse((results_temp$Weight > 0), 'blue', 'red'))
axis(2, at=xx, labels=results_temp$VariableName, tick=FALSE, las=2, line=-0.3, ce
x.axis=0.6)
```



Now comes the fun part, let's run the GBM model different quantities of top variables and score it with our validation set.

```
head(results)
```

```
##   VariableName Weight
## 1           V1      0
## 2           V2      0
## 3           V3      0
## 4           V4      0
## 5           V5      0
## 6           V6      0
```

```
tail(results)
```

```
##      VariableName    Weight
## 3305        V3305  25.35917
## 2856        V2856  26.22840
## 905          V905  63.38267
## 558          V558  71.52484
## 3976        V3976  87.59372
## 3657        V3657 321.01019
```

```
# Let's start by modeling only top 2 variable
traindf_truncated <- traindf[, c(tail(results$VariableName,2), 'cluster')]

# caret requires a factor of non-numeric value
dim(traindf_truncated)
```

```
## [1] 1500    3
```

```
# we don't need parameters as we already know the best ones from previous trainCon
trol call
fitControl <- trainControl(method="none")

gbm_model <- train(cluster~., data=traindf_truncated,
     tuneGrid = expand.grid(n.trees = 150, interaction.depth = 3, shrinkage = 0.1,
n.minobsinnode=10),
     trControl=fitControl, method="gbm", metric='roc')
```

```
predictions <- predict(object=gbm_model, gisette_df_validate[,setdiff(names(traind
f_truncated), 'cluster')], type='raw')

head(predictions)
```

```
## [1] yes yes yes yes no  no
## Levels: no yes
```

```
# 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=predictions, obs=gisette_df_validate$cluster))
```

```
## Accuracy    Kappa
##   0.8326   0.6650
```

Not bad, accuracy of 83.26% with only 2 variables!! Much better than our small PCA set in the previous lecture.

Let's try larger amounts of the top variables and measure accuracy on the validation set. We'll try anything with a weight larger than 10 (turns out to be 20 variables):

```
head(results)
```

```
##   VariableName Weight
## 1           V1      0
## 2           V2      0
## 3           V3      0
## 4           V4      0
## 5           V5      0
## 6           V6      0
```

```
tail(results)
```

```
##      VariableName Weight
## 339          V339  29.00
## 4425        V4425  31.19
## 3305        V3305  59.57
## 558          V558  73.17
## 3976        V3976  79.18
## 3657        V3657 304.19
```

```
# Let's start by modeling with different results$Weight values
traindf_truncated <- traindf[, c(results$VariableName[results$Weight > 10], 'clust
er')]
dim(traindf_truncated)
```

```
## [1] 1500   20
```

```
fitControl <- trainControl(method="none")

gbm_model <- train(cluster~., data=traindf_truncated,
tuneGrid = expand.grid(n.trees = 150, interaction.depth = 3, shrinkage = 0.1, n.mi
nobsinnode=10),
trControl=fitControl, method="gbm", metric='roc')

predictions <- predict(object=gbm_model, gisette_df_validate[,setdiff(names(traind
f_truncated), 'cluster')], type='raw')

print(postResample(pred=predictions, obs=as.factor(gisette_df_validate$cluster)))
```

```
## Accuracy    Kappa
##   0.9356   0.8712
```

Nice! Accuracy of 93.56% with the top 20 variables, a bit better than with our PCA model. One more…

```
traindf_truncated <- traindf[, c(results$VariableName[results$Weight > 2], 'cluste
r')]
dim(traindf_truncated)
```

```
## [1] 1500   58
```

```
fitControl <- trainControl(method="none")

gbm_model <- train(cluster~., data=traindf_truncated,
tuneGrid = expand.grid(n.trees = 150, interaction.depth = 3, shrinkage = 0.1, n.mi
nobsinnode=10),
trControl=fitControl, method="gbm", metric='roc')
```

```
predictions <- predict(object=gbm_model, gisette_df_validate[,setdiff(names(traind
f_truncated), 'cluster')], type='raw')

print(postResample(pred=predictions, obs=as.factor(gisette_df_validate$cluster)))
```

```
## Accuracy    Kappa
##   0.9576   0.9152
```

Yowza! Accuracy 95.76%!!!