Homework 2 - Analysis of High-Dimensional-Data

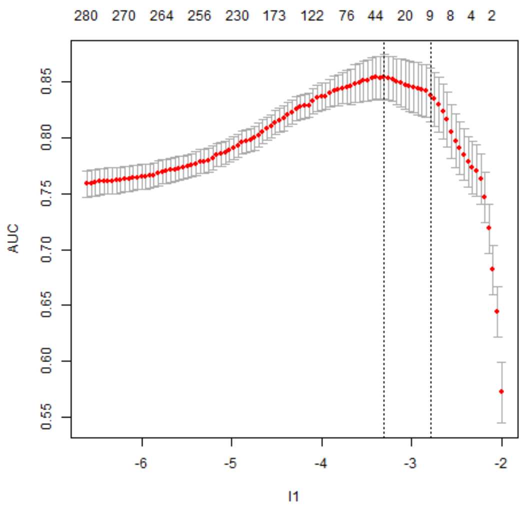
Brecht Dewilde

Model creation

In this homework a logistic regression model with lasso penalty term is created to classify a binary target variable. For this purpose a high dimensional dataset is used which contains 775 observations and 996 numeric predictors. Approximately 50% of the observations belong to class 1. Training data is sampled by randomly selecting 75% of the data. The remaining 25% will be used to assess the model performance. The l_1 penalty term is tuned by performing a 10-fold-cross validation grid search. The l_1 value that results in the highest cross-validation AUC metric is used to fit a first model. The coefficient estimates of this model are as follows:

```
    intercept           X22           X25           X31           X35
-0.0204762408  0.2760491612  0.4369205289  0.2653488061  0.3970760973
          X46           X86          X179          X317          X388
 0.0332901550  0.0571318048  0.0337040648  0.0001219754  0.0060698742
         X481          X562          X683          X714          X730
 0.0260960609  0.0023482482  0.0743771123  0.0435776828  0.2710133414
         X824
 0.0256810376
```

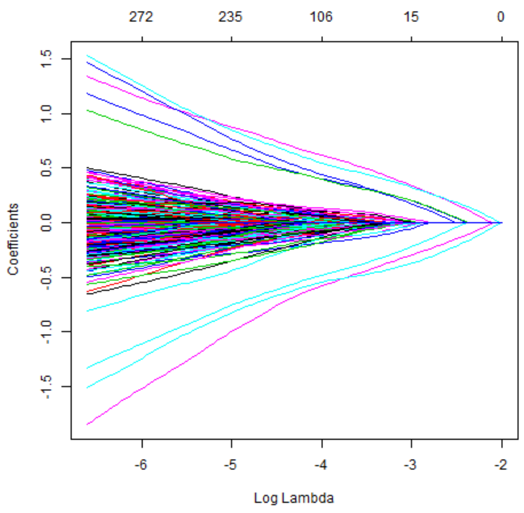
The objective of regularisation is to balance accuracy and simplicity. It can therefore be beneficial to find a l_1 value that lies within one standard error from the optimal value and results in simpler model (i.e. less predictors). A graphical representation of this approach is visualized below. The dashed lines indicate the range of l_1 values at one standard error from the optimal value. A logistic regression model is estimated for each of these value. The l_1 that results in the model with the least amount of predictors is chosen. The axis above this graph indicates the amount of predictors given a certain l_1 value. It is clear that the amount of predictors can be heavily reduced by ranging l_1 over this interval.



This approach leads to a model with only 7 predictors. This simplified model is considered as final. The model coefficient estimates are:

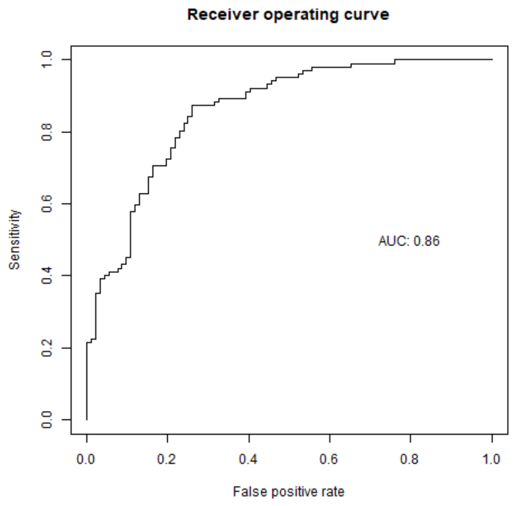
```
   intercept          X22          X25          X31          X35          X46 
-0.021408479  0.182249191  0.315359730  0.151198207  0.304848916  0.002992215 
        X683         X730 
 0.025299012  0.184802500 
```

The feature selection characteristic of the lasso penalty term is visualized in the figure below. Each line represents the coefficient estimate of a particular predictor for a given lambda value. The more the lambda value increases the more the coefficients shrink towards zero. The axis above of the graph shows the total number of predictor variables still included in the model.



Selecting the threshold value

The output of a logistic regression model is the estimated probability that the observation belongs to class 1: $\hat{\pi}(\boldsymbol{x}) = \mathrm{P}\{\widehat{Y = 1}|\boldsymbol{x}\}$. A classification threshold has to be defined, such that each observation with a predicted probability higher than this threshold will be assigned to class 1. A ROC curve of the final model is plotted to see the trade-off between the **true-positive rate** and the **false-positive rate** for different threshold values.



As there is no information about the target variable's meaning, it is is assumed that the cost of misclassifying a true positive is equal to the cost of misclassyfing a true negative. Consequently, there is no preference between the sensitivity(S_e) and specificity (S_p) of the model. In this case, there are generally two approaches to find a good threshold. The first approach is to set the threshold to that particular point where $S_e = S_p$. The second approach is to maximize ($S_e + S_p$). A summary of both methods with their resulting test set sensitivity and specificity is given below. The first threshold ($C = 0.49$) is chosen because the second threshold leads to a too dessimated sensitivity and specificity value.

```
threshold sensitivity specificity
0.4937526   0.7941176   0.7941176
0.4782727   0.8725490   0.7391304
```

R code

```r
## Load the dataset
load(file = "DataSet.RData")

## 75-25 train-test split
# generating random ID's for split
set.seed(123)
smp_size <- floor(0.75 * nrow(DataSet))
train_ind <- sample(seq_len(nrow(DataSet)), size = smp_size)

# train test split
train <- DataSet[train_ind, ]
test <- DataSet[-train_ind, ]

# x y split
ytrain <- train$Y
xtrain <- train[,c(2:length(train))]
ytest <- test$Y
xtest <- test[, c(2:length(test))]

# The glm package expects a matrix and not a dataframe for its input values
xtrain <- as.matrix(xtrain)
xtest <- as.matrix(xtest)

# Model creation and AUC Cross-validation
cv_auc <- cv.glmnet(x = xtrain, y = ytrain, alpha = 1, family = "binomial", type.measure = "auc")

# Obtain the best l1 penality term (the penalty term that leads to the largest AUC)
best_lambda <- cv_auc$lambda.min

best_model <- glmnet(x = xtrain, y = ytrain, family = "binomial", lambda = best_lambda)
coefficients <- c(coef(best_model)[1,], coef(best_model)[,1][coef(best_model)[,1] > 0])
names(coefficients) <- c("intercept", names(coefficients)[2:length(coefficients)])
coefficients

# plot with se range and influence on AUC and predictors
plot(cv_auc, xlab = "l1")
```

```r
# Create model with lambda within one se and least predictors
lambda_1se <- cv_auc$lambda.1se
modelse <- glmnet(x = xtrain, y = ytrain, family = "binomial", lambda = lambda_1se)
coefficients <- c(coef(modelse)[1,], coef(modelse)[,1][coef(modelse)[,1] > 0])
names(coefficients) <- c("intercept", names(coefficients)[2:length(coefficients)])
coefficients

# plot shrinkage influence of l1 penalty term
model <- glmnet(x = xtrain, y = ytrain, family = "binomial")
plot(model, xvar= "lambda")
```

```r
# probability estimates for the xtest dataset
ypred <- predict(modelse, newx = xtest, type = "response")

# obtain standardized class for roc creation
```

```r
predictions <- prediction(ypred, ytest)

# Get auc
auc <- round(unlist(performance(predictions, "auc")@y.values),3)

# ROCR plot
performance_measures <- performance(predictions, "sens", "fpr")
plot(performance_measures, main = "Receiver operating curve") +
  text(0.8, 0.5,  paste("AUC:", auc))
```

```r
# Approach 1
prbe_threshold <- unlist(performance(predictions, "prbe")@x.values)
prbe_sens_spec <- unlist(performance(predictions, "prbe")@y.values)

# Approach 2
roc_obj <- roc(ytest, ypred[,1])
```

```r
max_threshold <- coords(roc_obj, "best", "threshold")$threshold
```

```r
max_specificity <- coords(roc_obj, "best", "threshold")$specificity
```

```r
max_sensitivity <- coords(roc_obj, "best", "threshold")$sensitivity
```

```r
# Comparison of both approaches
threshold <- c(prbe_threshold, max_threshold)
sensitivity <- c(prbe_sens_spec, max_sensitivity)
specificity <- c(prbe_sens_spec, max_specificity)
df <- data.frame(threshold, sensitivity, specificity)
df
```