Biostat273 Final Project

Hubert Jenq

12/9/2016

Data:

The Titanic dataset from kaggle.com will be analyzed. The dataset includes 891 passengers with 12 covariates that are: Passenger ID, Survived, Ticket class, name, sex, age, #siblings, #children/spouse, fare, cabin, port of where they embarked upon. The goal is to predict the passengers survival. Two features were generated. #family members was calculated by the sum of the siblings and children/spouses. The names of the passengers included different titles such as Mr., Miss, Sir, etc. so the titles were extracted and used as an additional covariate. The features that would not have an effect on the prediction of survival like passenger id (random) were dropped from the analysis

Method:

Random forest, logistic regression with LASSO, and SVM with a linear and radial kernel will be implemented. The dataset was first split into 80/20 training/testing to determine which method would yield the best results. In order to optimize the parameters of the models, 10-repeats of 10-fold cross validation was done while varying parameters for each specific model. In random forest, mtry was varied from 2-27. Logistic regression had its lambda varied. The SVM models had the C parameter optimized where C=inf is a hard margin svm likely to overfit and C=0 would result in an underfit model.

```
set.seed(1)

library("e1071")
library("caret")
library(knitr)

data=read.csv('train.csv')

#Split data into 80/20
trainindex=sample(1:nrow(data),round(.8*nrow(data)))
testindex=1:nrow(data)
testindex=testindex[-trainindex]

str(data)
```

```
## 'data.frame':    891 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : Factor w/ 891 levels "Abbing, Mr. Anthony",..: 109 191 358 277 16 559 520 629 417 58
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : Factor w/ 681 levels "110152","110413",..: 524 597 670 50 473 276 86 396 345 133 ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
```

```
##  $ Cabin      : Factor w/ 148 levels "","A10","A14",..: 1 83 1 57 1 1 131 1 1 1 ...
##  $ Embarked   : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
```

```
#Change numeric variables that should be factors into factors
makefactor <- c('Survived', 'Pclass', 'Sex', 'Embarked')
data[makefactor] <- lapply(data[makefactor], function(x) as.factor(x))

#Changes names to something useful for prediction like titles
titles <-  gsub("^.*, (.*?)\\..*$", "\\1", data$Name)
table(titles)
```

```
## titles
##         Capt          Col          Don           Dr     Jonkheer
##            1            2            1            7            1
##         Lady        Major       Master         Miss         Mlle
##            1            2           40          182            2
##          Mme           Mr          Mrs           Ms          Rev
##            1          517          125            1            6
##          Sir the Countess
##            1            1
```

```
data$names=as.factor(titles)

#Add family size as a variable.
data$famsize=data$SibSp+data$Parch

#Use average age as NA age

data$Age[is.na(data$Age)]=mean(data$Age[!is.na(data$Age)])
#SVM

# Setup for cross validation
ctrl <- trainControl(method="repeatedcv",
                     repeats=5,
  summaryFunction=twoClassSummary,
                     classProbs=TRUE)

datatrain=data[trainindex,]
datatest=data[testindex,]

#do not use passenger id, cabin, survived, name, or ticket
datatrain=datatrain[,-c(1,4,9,11)]
datatest=datatest[,-c(1,4,9,11)]

trainY=datatrain[,2]
testX=datatest[,-c(1,2,4,11)]
testX=datatest[,-c(1,2,4,11)]
testY=datatest[,2]


#myControl <- trainControl(
  #method = "cv",
  #number = 10,
  #repeats = 10,
  #verboseIter = TRUE
```

```
#)

#rf_model <- train(
#  Survived ~.,
#  tuneGrid = data.frame(mtry = c(1:15)),
#  data = datatrain,
#  method = "ranger",
#  trControl = myControl,
#  importance = 'impurity'
#)

#Random forest with varying mtry.
#rf_model <- train(
 # Survived ~.,
#  tuneLength = 20,
#  data = datatrain,
#  method = "ranger",
#  trControl = myControl,
#  importance = 'impurity'
#)


#C is the fitting parameter, when infinity then hard margin SVM, when near 0 the model may underfit.
#svm_radial <- train(
 # Survived ~.,
  #tuneLength = 10,
  #data = datatrain,
  #method = "svmRadial",
  #trControl = myControl
#)

#C is the fitting parameter, when infinity then hard margin SVM, when near 0 the model may underfit.
#svm_linear <- train(
#  Survived ~.,
#  tuneLength = 10,
#  data = datatrain,
#  method = "svmLinear",
#  trControl = myControl
#)

#logistic regression with LASSO
#glm_model <- train(
#  Survived ~.,
#  method = "glmnet",
#  tuneGrid = expand.grid(alpha = 0:1,
 #                        lambda = seq(0.0001, 1, length = 20)),
#  data = datatrain,
#  trControl = myControl
#)

glmpredict <- predict(glm_model, datatest)
rfpredict <- predict(rf_model,datatest)
svmlinearpredict<-predict(svm_linear,datatest)
```

```r
svmradialpredict<-predict(svm_radial,datatest)
```

SVM

Using the linear kernel we select a cost C=1 and have 281 support vectors. The radial kernel we get a cost of also 1 and 354 support vectors. The training error from the CV step show 0.1669 and 0.147 respectively. However, when used to predict the test set the linear and radial kernel gave training errors of 0.185 and 0.191 respectively. This is likely due to the overfitting from the radial kernel since it has 73 more support vectors in a dataset size of only 713. Both the SVM models could be overfitting due to the high number of support vectors compared to datapoints.

```r
print(svm_linear$finalModel)
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 297
##
## Objective Function Value : -286.1448
## Training error : 0.175316
```

```r
print(svm_radial$finalModel)
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 8
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.0123502033392328
##
## Number of Support Vectors : 413
##
## Objective Function Value : -1891.899
## Training error : 0.11641
```

```r
linearSVMerror=sum(svmlinearpredict!=datatest$Survived)/nrow(datatest)
linearSVMerror
```

```
## [1] 0.1853933
```

```r
radialSVMerror=sum(svmradialpredict!=datatest$Survived)/nrow(datatest)
radialSVMerror
```

```
## [1] 0.1573034
```

Logistic regression with LASSO.

Logistic regression with LASSO yielded the lambda=0.0001 best CV training error of 0.176. The prediction on the training set gave an error of 0.179. The variable importance was taken from the logistic model by looking at the absolute value of the coefficients. Here we see that the names variable introduced gave the best prediction.

```
glmerror=sum(glmpredict!=datatest$Survived)/nrow(datatest)
glmerror
```

```
## [1] 0.1853933
```

```
varImp(glm_model)
```

```
## glmnet variable importance
##
##   only 20 most important variables shown (out of 27)
##
##                    Overall
## namesDon           100.000
## namesJonkheer       92.358
## namesRev            84.187
## namesMs             77.477
## namesLady           71.769
## namesMaster         67.439
## Sexmale             64.403
## namesthe Countess   57.195
## namesMlle           54.831
## namesMme            47.561
## Pclass3             38.545
## namesMajor          21.664
## Pclass2             17.453
## namesDr             16.032
## EmbarkedS           10.357
## namesMr              7.966
## namesMrs             6.864
## famsize              6.417
## SibSp                4.316
## EmbarkedC            3.735
```

Random forest

The random forest model used 500 trees and had the best OOB error of 16.69% with an mtry of 3 out of the 27 variables. The best random forest model gave an error of 0.179 on the test set prediction. Variable importance was assesed by the effect of prediction when the parameter values were permuted. Here we see

```
print(rf_model$finalModel)
```

```
## Ranger result
##
## Call:
##  ranger(.outcome ~ ., data = x, mtry = param$mtry, write.forest = TRUE,      probability = classProb
##
## Type:                             Classification
## Number of trees:                  500
## Sample size:                      713
## Number of independent variables:  27
## Mtry:                             11
## Target node size:                 1
```

```
## Variable importance mode:         impurity
## OOB prediction error:             18.09 %
```

```
rferror=sum(rfpredict!=datatest$Survived)/nrow(datatest)
rferror
```

```
## [1] 0.07303371
```

```
varImp(rf_model)
```

```
## ranger variable importance
##
##   only 20 most important variables shown (out of 27)
##
##                Overall
## Fare          100.0000
## Age            81.2836
## Sexmale        77.1957
## namesMr        71.2858
## Pclass3        33.5206
## famsize        28.9311
## SibSp          17.5263
## namesMiss      14.0408
## Parch           8.7772
## namesMrs        8.0022
## EmbarkedS       7.9977
## Pclass2         7.5623
## namesMaster     5.7324
## EmbarkedC       5.0067
## EmbarkedQ       3.7321
## namesDr         1.3709
## namesRev        1.2951
## namesMajor      0.6568
## namesDon        0.5972
## namesJonkheer   0.1385
```

Results

We see that the four models have very similar predictive power based on the data. The logistic regression and RF have the same predictive accuracy but they put very different weights on the variable importance. I would choose the random forest model as the variable importance is interpretable and we do not have to worry about linearity conditions. Further improvement on the dataset can be done by imputing missing values and attempting to come up with new features such as determining which individuals are related in a family.

```
summary=data.frame(t(c(linearSVMerror,radialSVMerror,glmerror,rferror)))
colnames(summary)=c("Linear SVM","Radial SVM","Logistic /w LASSO","Random Forest")
rownames(summary)="Test set error"
kable(summary,caption="Model test error comparison")
```

Table 1: Model test error comparison

	Linear SVM	Radial SVM	Logistic /w LASSO	Random Forest
Test set error	0.1853933	0.1573034	0.1853933	0.0730337