CSC 529 Advanced Data Mining - Assignment 3

Akhil Kumar Ramasagaram

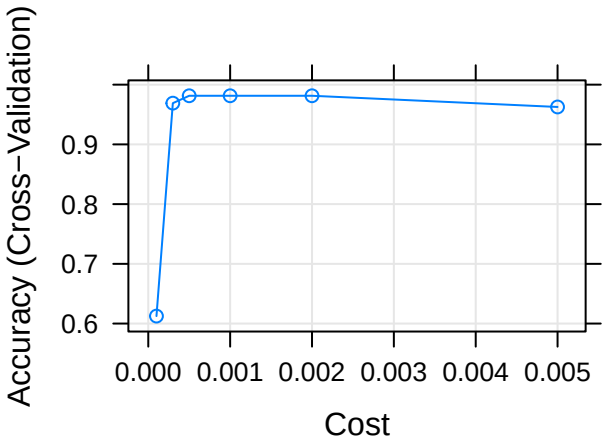
Tuesday, February 23, 2016

Problem 1 : (Handwriting recognition using support vector machines)

In this problem, you will apply a support vector machine to classify hand-written digits. Download the digit data set from the course documents for week 7. The zip archive contains two text files. The file uspsdata.txt contains a matrix with one digit/data point (= vector of length 256) per row. The 256-vector in each row represents a 16 by 16 image of a handwritten number. The file uspscl.txt contains the corresponding class labels. The data contains two classes, the digits 5 and 6, and the class labels are stored as -1 and +1, respectively.

a) Train a linear SVM with soft margin. Vary the soft margin parameter and plot the classification error as a function of the margin parameter. Discuss the results.

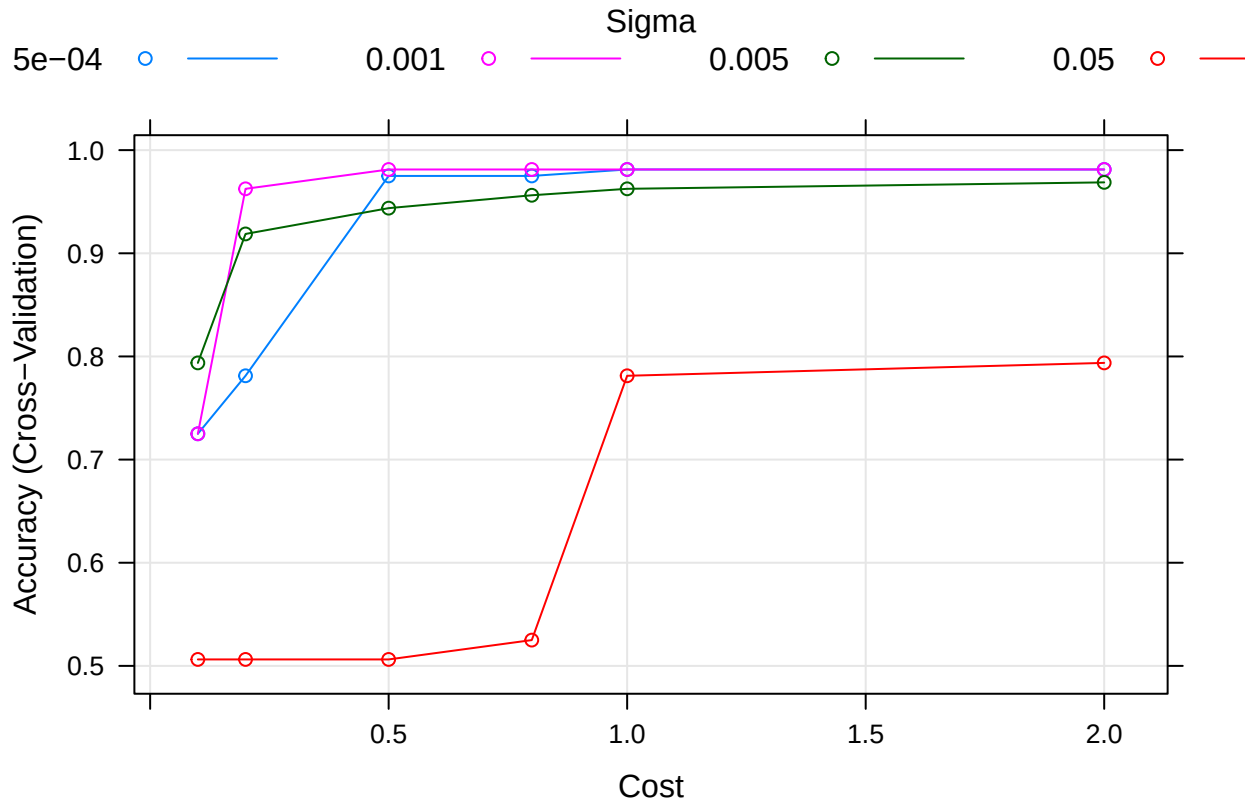
```
library(caret)
library(e1071)
hr <- read.csv("uspsdata.txt", header = F, sep = "")
hr_label <- read.table("uspscl.txt", header = F)$V1

ind <- sample(1:200,200*.8,F)
train <- hr[ind,]
train_label <- hr_label[ind]
test <- hr[-ind,]
test_label <- hr_label[-ind]

svm_ln <- train(train,as.factor(train_label),
             method = "svmLinear",
             tuneGrid = expand.grid(C = c(0.0001,0.0003,0.0005,0.001,0.002,0.005)),
             metric = "Accuracy",
             trControl = trainControl(method = "cv",
                                      number = 3))
plot(svm_ln)
```



The optimal soft margin was some where between 0 - 0.0001 and after that the error converged.

b) Train a non-linear SVM with soft margin and Gaussian kernel. Vary both the soft margin parameter and the Gaussian kernel bandwidth (sigma) and plot the classification error as a function of the margin parameter and kernel bandwidth.

```
svmGrid <- expand.grid(sigma= c(0.0005,0.005,0.001,0.05), C= c(0.1,0.2,0.5,0.8,1,2))
set.seed(45)
svm_nl <- train(train,as.factor(train_label),
                method = "svmRadial",
                tuneGrid = svmGrid,
                metric = "Accuracy",
                trControl = trainControl(method = "cv",
                                         number = 2))
plot(svm_nl)
```



c) After you have selected parameter values for both algorithms (in part a. and part b.), and trained each one with the parameter value you have chosen, compute the classification error on the test set. Report the test set estimates of the error for both cases along with the parameter values you have selected, and compare the two results. Is a linear SVM a good choice for this data, or should we use a non-linear one?

```
confusionMatrix(predict(svm_ln$finalModel, test),test_label)$overall[1]
```

```
## Accuracy
##    0.925
```

```
confusionMatrix(predict(svm_nl$finalModel, test),test_label)$overall[1]
```

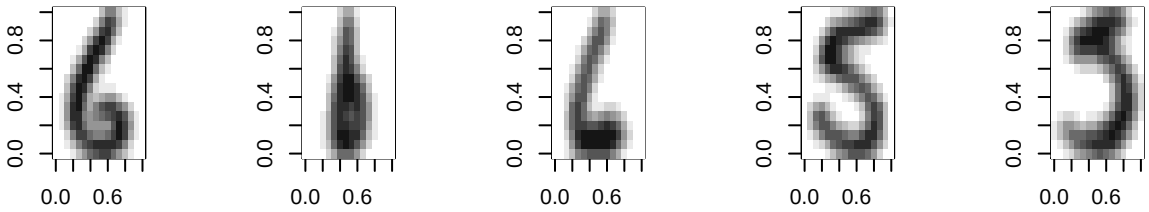
```
## Accuracy
##     0.95
```

The non linear model performed much better than linear model. d)) For each method (linear and Gaussian), select five correctly and five incorrectly classified samples. Plot these as images like the one above.

```
show_digit <- function(data, ind, col = gray(12:1 / 12)) {
    image(matrix(as.numeric(data[ind, 1:256]), nrow = 16)[, 16:1], col = col)
}
ln_correct <- which(predict(svm_ln, test) == 1 & test_label == 1)
ln_correct <- sample(1:length(ln_correct), 5, F)
ln_incorrect <- which(predict(svm_ln, test) == 1 & test_label == -1)
nl_correct <- which(predict(svm_nl, test) == 1 & test_label == 1)
nl_correct <- sample(1:length(nl_correct), 5, F)
nl_incorrect <- which(predict(svm_nl, test) == 1 & test_label == -1)
```

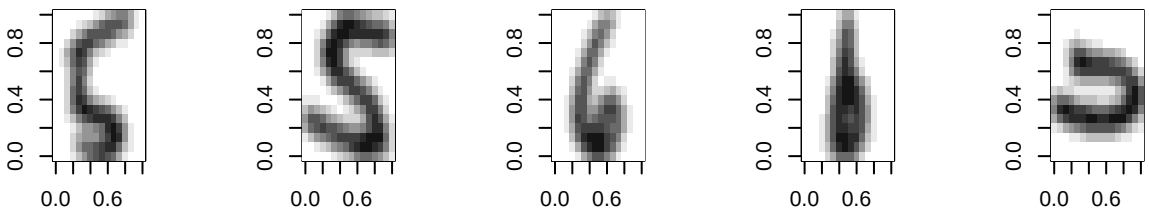
plotting 5 random correct prediction using linear svm

```
par(mfrow = c(1,5))
for(i in 1:5) print(show_digit(test,ln_correct[i]))
```



plotting 5 random correct prediction using non_linear svm

```
par(mfrow = c(1,5))
for(i in 1:5) print(show_digit(test,nl_correct[i]))
```



plotting the single misclassified observation using linear & non linear svm

```
par(mfrow = c(2,3))
for(i in c(ln_incorrect,nl_incorrect)) print(show_digit(test,i))
```

Problem 2: (E-commerce Customer Identification using ensemble of classifiers)

The data of e-tailer customers is posted under the course documents for week 7. The training data contains 334 variables for a known set of 10,000 customers and non-customers with a ratio of 1:10, respectively. The test data consists of a set of examples and is drawn from the same distribution as the training set. The feature data is train10000.csv and the label data is train10000_label.csv with corresponding labels for the records in train10000.csv. The test10000.csv is the test data with corresponding labels for the records in test10000-label.csv.

```
train_data <- read.csv("assignment3_problem2/train10000.csv", heade = F)
train_data[train_data == 999000] <- NA
cs <- colSums(train_data)
for(i in 1:ncol(train_data)){
  if(is.na(cs[i])){
    train_data[is.na(train_data[,i]),i] <- mean(na.omit(train_data[,i]))
  }
}
train_label <- read.csv("assignment3_problem2/train10000_Label.csv", header = F)
train_data$Target <- as.factor(train_label$V1)
```

```
library(randomForest)
library(rpart)
prb_2 <- function(data,model){
  model_result <- data.frame("data" = NULL, "sensitivity" = NULL, "specificity" = NULL)
  print("Using Raw Data")
  if(model == "dt"){
    raw_model <- rpart(Target ~ ., data)
  }
  else{
    raw_model <- randomForest(Target ~ ., data)
```

```
  }
  pred <- predict(raw_model, newdata = train_data, type = "class")
  cf <- confusionMatrix(data = pred,reference = data$Target)
  model_result <- rbind(model_result, data.frame("data" = "Raw Data", "sensitivity" = cf$byClass[1],"sp
  # balance class
  print("Balancing Data")
  if(model == 'dt'){
    positive_class <- data[data$Target == 1,]
    negative_class <- data[data$Target == 0,]
    negative_class <- negative_class[sample(1:nrow(negative_class),nrow(positive_class)),]
    balance_data <- rbind(positive_class,negative_class)
    balance_model <- rpart(Target ~ ., balance_data)
    pred <- predict(balance_model, newdata = balance_data, type = "class")
    cf <- confusionMatrix(data = pred,reference = balance_data$Target)
  }
  else{
    balance_model <- randomForest(Target ~., data = data, strata = data$Target, sampsize = rep(sum(data$
    pred <- predict(balance_model, newdata = data, type = "class")
    cf <- confusionMatrix(data = pred,reference = data$Target)
  }
  model_result <- rbind(model_result, data.frame("data" = "Balance Data", "sensitivity" = cf$byClass[1]
  #normalize
  print("Normalizing")
  normalize_data <- data.frame(scale(data[,!names(data) %in% "Target"]))
  normalize_data$Target <- data$Target
  if(model == "dt"){
    norm_model <-  rpart(Target ~ ., normalize_data)
  }
  else{
    norm_model <-  randomForest(Target ~ ., normalize_data)
  }
  pred <- predict(norm_model, newdata = normalize_data, type = "class")
  cf <- confusionMatrix(data = pred,reference = normalize_data$Target)
  model_result <- rbind(model_result, data.frame("data" = "Normalized Data", "sensitivity" = cf$byClass
  #feature_selection
  print("Feature Selection")
  pc_data <- prcomp(data[,!names(data) == "Target"])
  pc_data <- data.frame(pc_data$x)
  pc_data <- data.frame("PC1" = pc_data$PC1, "PC2" = pc_data$PC2, "Target" = data$Target)
  if(model == "dt"){
    pc_model <- rpart(Target ~ ., pc_data)
  }
  else{
    pc_model <- randomForest(Target ~ ., pc_data)
  }
  pred <- predict(pc_model, newdata = pc_data, type = "class")
  cf <- confusionMatrix(data = pred,reference = pc_data$Target)
  model_result <- rbind(model_result, data.frame("data" = "Feature Selection", "sensitivity" = cf$byCla
  print(model_result)
  all_models <- list(raw_model, balance_model, norm_model, pc_model)
#   best_model <- all_models[[which.max(model_result[,2] + model_result[,3])]]
  return(all_models)
}
```

```
best_dt <- prb_2(train_data,"dt")
```

```
## [1] "Using Raw Data"
## [1] "Balancing Data"
## [1] "Normalizing"
## [1] "Feature Selection"
##                            data sensitivity specificity
## Sensitivity           Raw Data   1.0000000   0.0000000
## Sensitivity1      Balance Data   0.7161716   0.5731573
## Sensitivity2   Normalized Data   1.0000000   0.0000000
## Sensitivity3 Feature Selection   1.0000000   0.0000000
```

```
best_rf <- prb_2(train_data,"rf")
```

```
## [1] "Using Raw Data"
## [1] "Balancing Data"
## [1] "Normalizing"
## [1] "Feature Selection"
##                            data sensitivity specificity
## Sensitivity           Raw Data   1.0000000   0.8976898
## Sensitivity1      Balance Data   0.8750412   0.9823982
## Sensitivity2   Normalized Data   0.9998900   0.8921892
## Sensitivity3 Feature Selection   1.0000000   0.9823982
```

Testing

```
test_data <- read.csv("assignment3_problem2/test10000.csv", heade = F)
test_data[test_data == 999000] <- NA
cs <- colSums(test_data)
for(i in 1:ncol(test_data)){
  if(is.na(cs[i])){
    test_data[is.na(test_data[,i]),i] <- mean(na.omit(test_data[,i]))
  }
}
test_label <- read.csv("assignment3_problem2/test10000_label.csv", header = F)
test_data$Target <- as.factor(test_label$V1)
```

For missing values i substituted them with the average value and since the data contains features where values range for each features. In order to prevent larger values get overweighted i had to normalize all features so that they range from 0-1. For feature selection, i used the PCA to pick most significant components. so if you look at the results the RF model which was trained on balanced data was better.

```
test_result <- function(test, model){
  model_result <- data.frame("data" = NULL, "sensitivity" = NULL, "specificity" = NULL)
  #raw data
  pred <- predict(model[[1]], newdata = test, type = "class")
  cf <- confusionMatrix(data = pred,reference = test$Target)
  model_result <- rbind(model_result, data.frame("data" = "Raw Data", "sensitivity" = cf$byClass[1],"sp

  ##balance data
```

```
  pred <- predict(model[[2]], newdata = test, type = "class")
  cf <- confusionMatrix(data = pred,reference = test$Target)
  model_result <- rbind(model_result, data.frame("data" = "Balance Data", "sensitivity" = cf$byClass[1]

  #normalized data

  normalize_data <- data.frame(scale(test[,!names(test) %in% "Target"]))
  normalize_data$Target <- test$Target
  pred <- predict(model[[3]], newdata = normalize_data, type = "class")
  cf <- confusionMatrix(data = pred,reference = test$Target)
  model_result <- rbind(model_result, data.frame("data" = "Balance Data", "sensitivity" = cf$byClass[1]

  # normalized data
  pc_data <- prcomp(test[,!names(test) == "Target"])
  pc_data <- data.frame(pc_data$x)
  pc_data <- data.frame("PC1" = pc_data$PC1, "PC2" = pc_data$PC2, "Target" = test$Target)
  pred <- predict(model[[4]], newdata = pc_data, type = "class")
  cf <- confusionMatrix(data = pred,reference = test$Target)
  model_result <- rbind(model_result, data.frame("data" = "Feature Selection", "sensitivity" = cf$byCla
  return(model_result)
}

test_result(test_data, best_dt)
```

```
##                          data sensitivity specificity
## Sensitivity           Raw Data    1.000000   0.0000000
## Sensitivity1      Balance Data    0.636464   0.5537806
## Sensitivity2      Balance Data    1.000000   0.0000000
## Sensitivity3 Feature Selection    1.000000   0.0000000
```

```
test_result(test_data, best_rf)
```

```
##                          data sensitivity specificity
## Sensitivity           Raw Data   0.9822315  0.02023429
## Sensitivity1      Balance Data   0.8095133  0.34398296
## Sensitivity2      Balance Data   0.9913917  0.01490948
## Sensitivity3 Feature Selection   1.0000000  0.00000000
```