Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering
IDP Institut für Datenanalyse
und Prozessdesign

HS 2016

Statistisches Data Mining (StDM)

Woche 10

Aufgabe 1 Lab

Read and do the excersises of section 9.6.1 and 9.6.2 in ISLR.

Aufgabe 2 SVM for orange juice

This excerise is taken from ISLR:

a) Use the OJ data and create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

```
library(ISLR)
library(e1071)
set.seed(9004)
train = sample(dim(OJ)[1], 800)
OJ.train = OJ[train, ]
OJ.test = OJ[-train, ]
```

b) Fit a support vector classifier to the training data using `cost=0.01`, with Purchase as the response and the other variables as predictors. Use the `summary()` function to produce summary statistics, and describe the results obtained.

```
svm.linear = svm(Purchase~., kernel="linear", data=OJ.train, cost=0.01)
summary(svm.linear)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "linear",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##       gamma:  0.05555556
##
## Number of Support Vectors:  432
##
##  ( 217 215 )
##
##
```

```
## Number of Classes:  2
##
## Levels:
##  CH MM
```

Support vector classifier creates 432 support vectors out of 800 training points. Out of these, 217 belong to level `CH` and remaining 215 belong to level `MM`.

c) What are the training and test error rates?

```
  train.pred = predict(svm.linear, OJ.train)
  sum(OJ.train$Purchase != train.pred) / nrow(OJ.train)
```

```
## [1] 0.16875
```

```
  test.pred = predict(svm.linear, OJ.test)
  sum(OJ.test$Purchase != test.pred) / nrow(OJ.test)
```

```
## [1] 0.1777778
```

d) Use the `tune()` function to select an optimal cost. Consider val- ues in the range 0.01 to 10.

```
  set.seed(1554)
  tune.out = tune(svm, Purchase~., data=OJ.train, kernel="linear", ranges=list(cost=10^seq(-2, 1, by
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##       cost
##  0.3162278
##
## - best performance: 0.16875
##
## - Detailed performance results:
##           cost   error dispersion
## 1   0.01000000 0.16875 0.03691676
## 2   0.01778279 0.16875 0.03397814
## 3   0.03162278 0.17125 0.03230175
## 4   0.05623413 0.17250 0.03162278
## 5   0.10000000 0.17000 0.03291403
## 6   0.17782794 0.17125 0.03335936
## 7   0.31622777 0.16875 0.03498512
## 8   0.56234133 0.17000 0.03129164
## 9   1.00000000 0.16875 0.03397814
## 10  1.77827941 0.16875 0.03240906
## 11  3.16227766 0.16875 0.03294039
## 12  5.62341325 0.17125 0.03120831
## 13 10.00000000 0.17125 0.03283481
```

e) Compute the training and test error rates using this new value for `cost`.

```
  svm.linear = svm(Purchase~., kernel="linear", data=OJ.train, cost=tune.out$best.parameters$cost)
  train.pred = predict(svm.linear, OJ.train)
  sum(OJ.train$Purchase != train.pred) / nrow(OJ.train)
```

```
## [1] 0.16
```

```
  test.pred = predict(svm.linear, OJ.test)
  sum(OJ.test$Purchase != test.pred) / nrow(OJ.test)
```

```
## [1] 0.1814815
```