Stat 432 Homework 10

Giang Le

Assigned: Nov 1, 2021; Due: 11:59 PM CT, Nov 9, 2021

Contents

Question 1: Linear SVM and support vectors

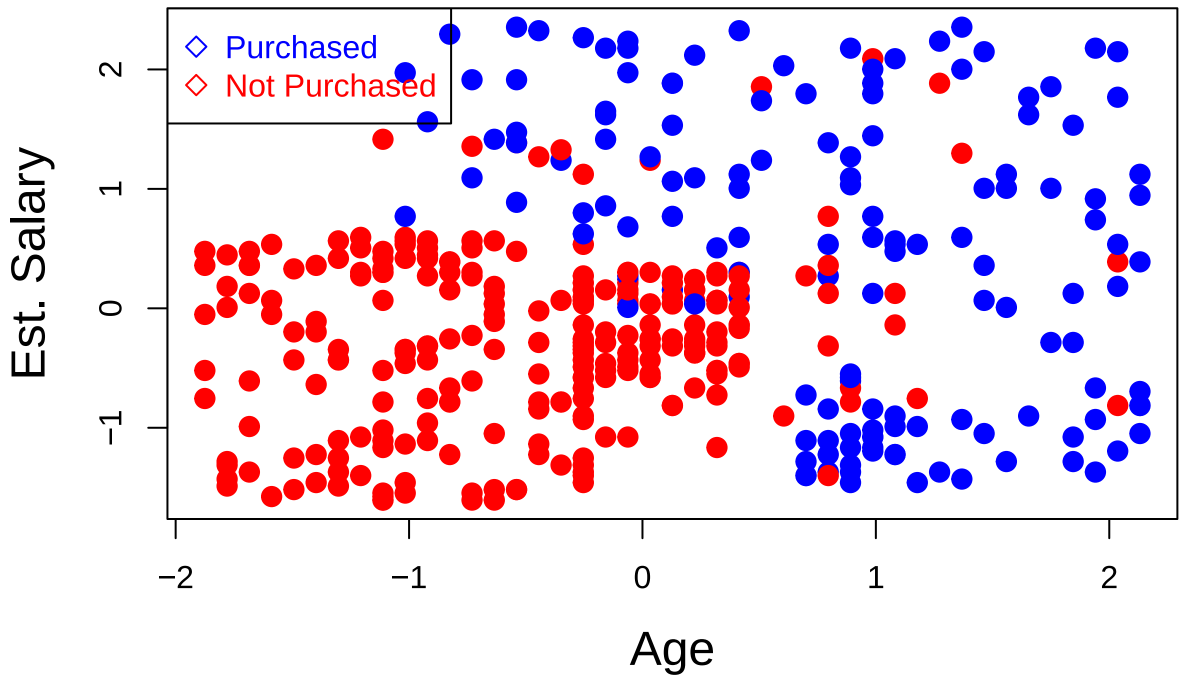
We will use the `Social Network Ads` data, available on Kaggle [link]. The `.csv` file is also available at our course website. The goal is to classify the outcome `Purchased`, and we will only use the two continuous variables `EstimatedSalary` and `Age`. **Scale and center both covariates before you proceed with the analysis**. For this question, you should use the `e1071` package. Complete the following tasks:

- [5 Points] Produce a 2d scatter plot of the data, with each observation colored by the outcome. Use `pch = 19` for the dots.

```
# Read in the data.
set.seed(662095561)
library(e1071)
sn <- read.csv("Social_Network_Ads.csv")
sn_data <- data.matrix(sn[,c(3:5)])
X <- sn_data[,c(1,2)]

# Scale and center the covariates.
center_scale <- function(x) {
    scale(x)
}

X <- center_scale(X)
y <- sn_data[,c(3)]
# plot
plot(X,col=ifelse(y>0,"blue","red"), pch = 19, cex = 1.2, lwd = 2,
         xlab = "Age", ylab = "Est. Salary", cex.lab = 1.5)
legend("topleft", c("Purchased", "Not Purchased"),col=c("blue", "red"),
           pch=c(5, 5), text.col=c("blue", "red"), cex = 1)
```



* [10 Points] Fit a linear SVM with `cost = 1`. Do not scale or center the data.

```
svm.fit <- svm(y ~ X, type='C-classification',
                   kernel='linear', scale=FALSE, cost = 1)
summary(svm.fit)
```

```
##
## Call:
## svm(formula = y ~ X, type = "C-classification", kernel = "linear",
##     cost = 1, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  157
##
##  ( 78 79 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

* [10 Points] What is the training data (in-sample) classification error? Also provide a confusion table of the results.

```
svm.pred <- predict(svm.fit, sn_data)
confusion <- table(true=y,svm.pred)

# Confusion matrix:
```

```r
confusion
```

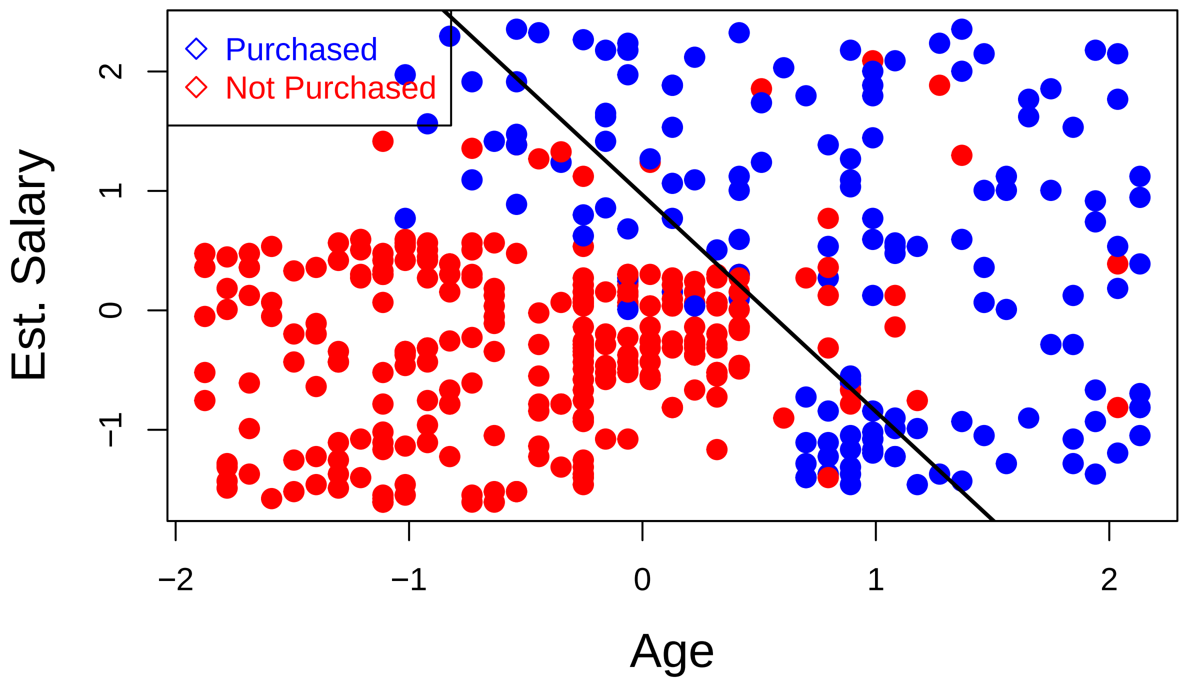
```
##      svm.pred
## true    0   1
##     0 240  17
##     1  46  97
```

```r
# Classification error
1-sum(diag(confusion))/length(y)
```

```
## [1] 0.1575
```

- [15 Points] Draw the decision line on the plot. For this question, you should try to use the `coefs`, `SV` and the `rho` from the fitted object, and calculate $\boldsymbol{\beta}$ and β_0. Note that the decision line is $f(x) = x^T\boldsymbol{\beta} + \beta_0 = 0$, you calculate the decision line based on them. An example can be found in the lecture note.

```r
b <- t(svm.fit$coefs) %*% svm.fit$SV
b0 <- -svm.fit$rho
plot(X,col=ifelse(y>0,"blue","red"), pch = 19, cex = 1.2, lwd = 2,
         xlab = "Age", ylab = "Est. Salary", cex.lab = 1.5)
legend("topleft", c("Purchased", "Not Purchased"),col=c("blue", "red"),
           pch=c(5, 5), text.col=c("blue", "red"), cex = 1)
abline(a= -b0/b[1,2], b=-b[1,1]/b[1,2], col="black", lty=1, lwd = 2)
```

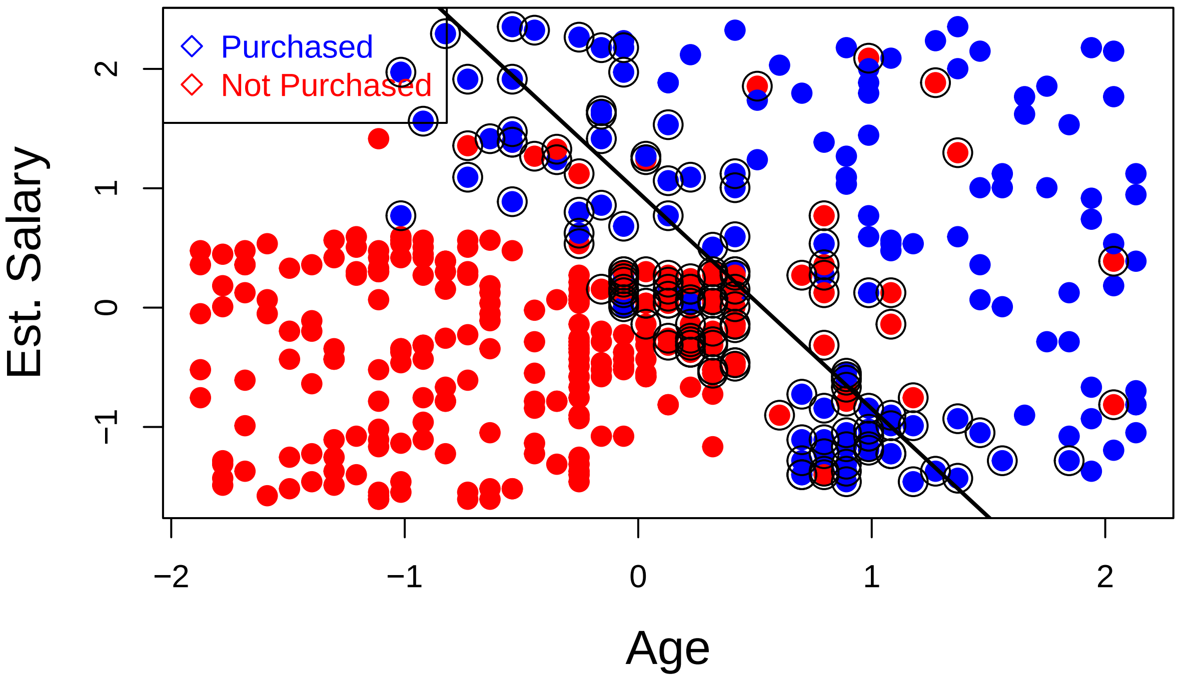


* [10 Points] Mark the support vectors on the plot (with a circle on the observation, use `cex = 2`).

```r
b <- t(svm.fit$coefs) %*% svm.fit$SV
b0 <- -svm.fit$rho
plot(X,col=ifelse(y>0,"blue","red"), pch = 19, cex = 1.2, lwd = 2,
         xlab = "Age", ylab = "Est. Salary", cex.lab = 1.5)
legend("topleft", c("Purchased", "Not Purchased"),col=c("blue", "red"),
           pch=c(5, 5), text.col=c("blue", "red"), cex = 1)
abline(a= -b0/b[1,2], b=-b[1,1]/b[1,2], col="black", lty=1, lwd = 2)

# Mark the support vectors
```

```r
points(X[svm.fit$index, ], col="black", cex=2)
```



Question 2: SVM for hand written digit Data

Take digits 4 and 9 from `zip.train` and `zip.test` in the `ElemStatLearn` library. For this question, you should use the `kernlab` package, in combination with the `caret` package to tune the parameters. Make sure that you specify the `method` argument so that the correct package/function is used to fit the model. You may consider reading the details from this documentation. Complete the following task.

- [5 Points] Construct the training and testing data so that they become a binary classification problem.

```r
install.packages("kernlab",repos = "http://cran.us.r-project.org")
```

```
##
## The downloaded binary packages are in
##  /var/folders/9c/3_mgdyf12z7dvb8rt4d60nt80000gn/T//Rtmpa5mufI/downloaded_packages
```

```r
library(kernlab)
library(caret)
```

```
## Loading required package: ggplot2
##
## Attaching package: 'ggplot2'
## The following object is masked from 'package:kernlab':
##
##     alpha
## Loading required package: lattice
```

```r
# Load data from my local dir.
load("/Users/gianghale/Desktop/fall-2021/stat-432/ElemStatLearn/data/zip.train.RData")
load("/Users/gianghale/Desktop/fall-2021/stat-432/ElemStatLearn/data/zip.test.RData")
```

```r
# Look at the dimensions of train and test data
dim(zip.train)
```

```
## [1] 7291  257
```

```r
## [1] 7291  257
dim(zip.test)
```

```
## [1] 2007  257
```

```r
## [1] 2007  257

df.train <- data.frame(zip.train)
df.test <- data.frame(zip.test)

# I use subset to select only columns where values are 4 and 9 for the digits.

df.train.filtered <- subset(df.train, df.train$X1 == 4 | df.train$X1 == 9)
df.train.filtered$X1 <- as.factor(df.train.filtered$X1)
df.test.filtered <- subset(df.test, df.test$X1 == 4 | df.test$X1 == 9)
df.test.filtered$X1 <- as.factor(df.test.filtered$X1)
```

- [15 Points] Construct a grid of tuning parameters for linear SVM using the `kernlab` package, and tune this using `caret`. Use 10-fold cross-validation for this question. What is the best `C` you obtained based on the accuracy? Predict the testing data using this model and obtain the confusion table and testing data accuracy.

```r
library(caret)

cost.grid = expand.grid(cost = seq(0.01, 2, length = 20))
train_control = trainControl(method="cv", number=10)

names(df.train.filtered)[1] <- "digits"
names(df.train.filtered)[2:257] <- paste(rep("factor",256), seq(2:257))
names(df.test.filtered)[1] <- "digits"
names(df.test.filtered)[2:257] <- paste(rep("factor",256), seq(2:257))

svm2 <- train(digits ~ ., data = df.train.filtered, method = "svmLinear2",
                 trControl = train_control,
                tuneGrid = cost.grid)
svm2
```

```
## Support Vector Machines with Linear Kernel
##
## 1296 samples
##  256 predictor
##    2 classes: '4', '9'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1166, 1167, 1166, 1167, 1166, 1167, ...
## Resampling results across tuning parameters:
##
##   cost       Accuracy   Kappa
##   0.0100000  0.9899702  0.9799405
##   0.1147368  0.9853369  0.9706727
```

```
##   0.2194737  0.9814729  0.9629431
##   0.3242105  0.9799284  0.9598532
##   0.4289474  0.9799344  0.9598658
##   0.5336842  0.9791592  0.9583163
##   0.6384211  0.9791592  0.9583163
##   0.7431579  0.9791592  0.9583163
##   0.8478947  0.9791592  0.9583163
##   0.9526316  0.9791592  0.9583163
##   1.0573684  0.9791592  0.9583163
##   1.1621053  0.9791592  0.9583163
##   1.2668421  0.9791592  0.9583163
##   1.3715789  0.9791592  0.9583163
##   1.4763158  0.9791592  0.9583163
##   1.5810526  0.9791592  0.9583163
##   1.6857895  0.9791592  0.9583163
##   1.7905263  0.9791592  0.9583163
##   1.8952632  0.9791592  0.9583163
##   2.0000000  0.9791592  0.9583163
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cost = 0.01.
```

The best c value contained is 0.01.

```r
# Calculate classification error on the test data and confusion matrix.

svm2.pred <- predict(svm2, df.test.filtered)
confusion2 <- table(true=df.test.filtered$digits,svm2.pred)

# Confusion matrix:
confusion2
```

```
##     svm2.pred
## true   4   9
##    4 192   8
##    9   5 172
```

```r
# Classification error
1-sum(diag(confusion2))/length(df.test.filtered$digits)
```

```
## [1] 0.03448276
```

```r
# Accuracy (96.55%)
sum(diag(confusion2))/length(df.test.filtered$digits)
```

```
## [1] 0.9655172
```

- [20 Points] Construct a grid of tuning parameters for radial Kernel SVM using the `kernlab` package, and tune this using `caret`. Use 10-fold cross-validation for this question. You may need to try this a few time to get a good range of tuning parameter. What is the best `C` and `sigma` you obtained based on the accuracy? Predict the testing data using this model and obtain the confusion table and testing data accuracy.

```r
svm.radial <- train(digits ~ ., data = df.train.filtered, method = "svmRadial",
                preProcess = c("center", "scale"),
                tuneGrid = expand.grid(C = c(0.001, 0.01, 0.1, 0.2), sigma = c(0.05, 0.01, 1)),
                trControl = trainControl(method = "cv", number = 10))
svm.radial
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 1296 samples
##  256 predictor
##    2 classes: '4', '9'
##
## Pre-processing: centered (256), scaled (256)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1166, 1166, 1167, 1166, 1167, 1167, ...
## Resampling results across tuning parameters:
##
##   C      sigma  Accuracy   Kappa
##   0.001  0.01   0.5030889  0.000000000
##   0.001  0.05   0.5030889  0.000000000
##   0.001  1.00   0.5030889  0.000000000
##   0.010  0.01   0.5363031  0.067239717
##   0.010  0.05   0.5030889  0.000000000
##   0.010  1.00   0.5030889  0.000000000
##   0.100  0.01   0.9089792  0.817770068
##   0.100  0.05   0.5046274  0.003076923
##   0.100  1.00   0.5030889  0.000000000
##   0.200  0.01   0.9352228  0.870350739
##   0.200  0.05   0.5756417  0.146689322
##   0.200  1.00   0.5030889  0.000000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.01 and C = 0.2.
```

The best C obtained was 0.2, sigma 0.01

```r
# Calculate classification error on the test data and confusion matrix.
svm3.pred <- predict(svm.radial, df.test.filtered)
confusion3 <- table(true=df.test.filtered$digits,svm3.pred)

# Confusion matrix:
confusion3
```

```
##     svm3.pred
## true   4   9
##    4 198   2
##    9  22 155
```

```r
# Classification error
1-sum(diag(confusion3))/length(df.test.filtered$digits)
```

```
## [1] 0.06366048
```

```r
# Accuracy (93.63%)
sum(diag(confusion3))/length(df.test.filtered$digits)
```

```
## [1] 0.9363395
```