Stat 432 Homework 10

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.

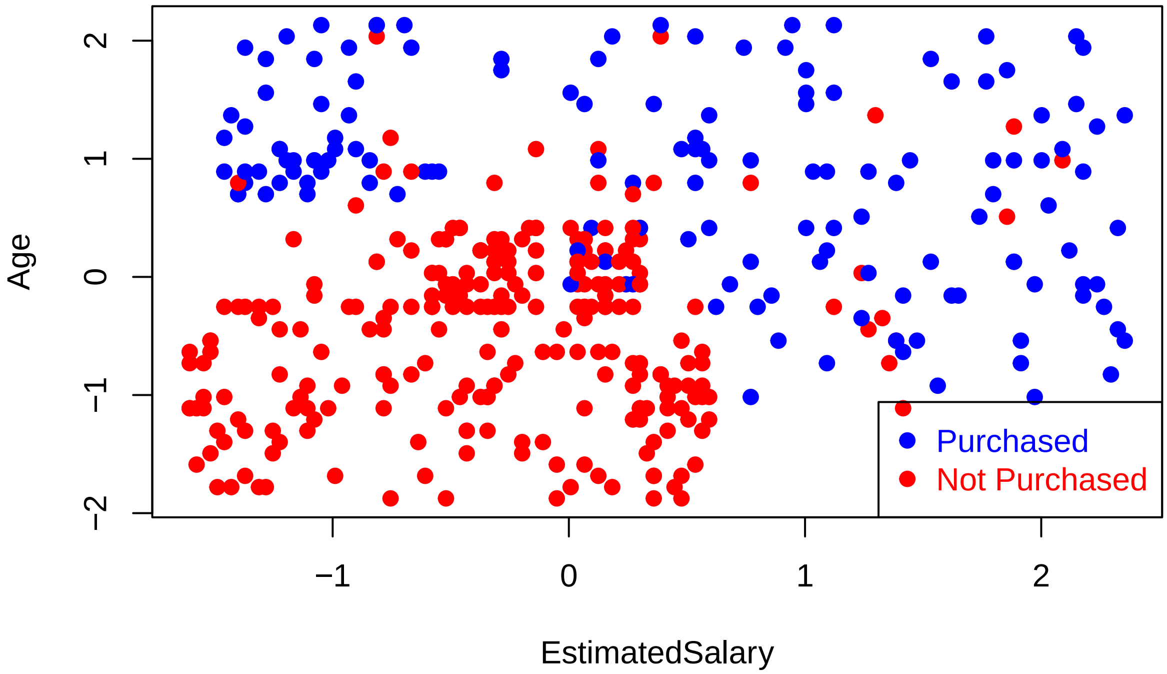
```r
# Loading library
library(e1071)
# Loading Social_Network_Ads data
data <- read.csv("Social_Network_Ads.csv")
# Creating data frame of 2 vars- 'EstimatedSalary', 'Age'
unscaled_vars <- data[, c("EstimatedSalary", "Age")]


# Scaling and centering both variables
scaled_vars <- data.frame(scale(unscaled_vars))

# Converting class 0 to -1
data$Purchased[which(data$Purchased==0)]=-1

# 2d scatter plot
plot(
  scaled_vars,
  col = ifelse(data$Purchased > 0, "blue", "red"),
  pch = 19,                                    # Using `pch = 19` for the dots
  xlab = "EstimatedSalary",
  ylab = "Age"
)
legend(
  "bottomright",
  c("Purchased", "Not Purchased"),
  col = c("blue", "red"),
  pch = c(19, 19),
```

```
  text.col = c("blue", "red")
)
```



The plot is a 2d scatter plot of the data with each observation colored by the outcome.

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

```
# Creating a data frame of both scaled variables and 'Purchased'
use_data <- scaled_vars
use_data$Purchased <- data$Purchased


# Fitting a linear SVM with `cost = 1`
svm.fit <-
  svm(
    Purchased ~ .,
    data = use_data,
    type = 'C-classification',
    kernel = 'linear',
    scale = FALSE,
    cost = 1
  )
```

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

```
# Making predictions using linear SVM
Y.pred = predict(svm.fit, scaled_vars)

# Confusion table of results
confusion_table_1=table(use_data$Purchased, Y.pred)
confusion_table_1
```

```
##     Y.pred
##       -1   1
##   -1 240  17
##   1   46  97
```

```
# In-sample classification error
100*(confusion_table_1[1,2]+confusion_table_1[2,1])/length(Y.pred)
```

```
## [1] 15.75
```

The in-sample classification error using the linear SVM model is 15.75%.

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

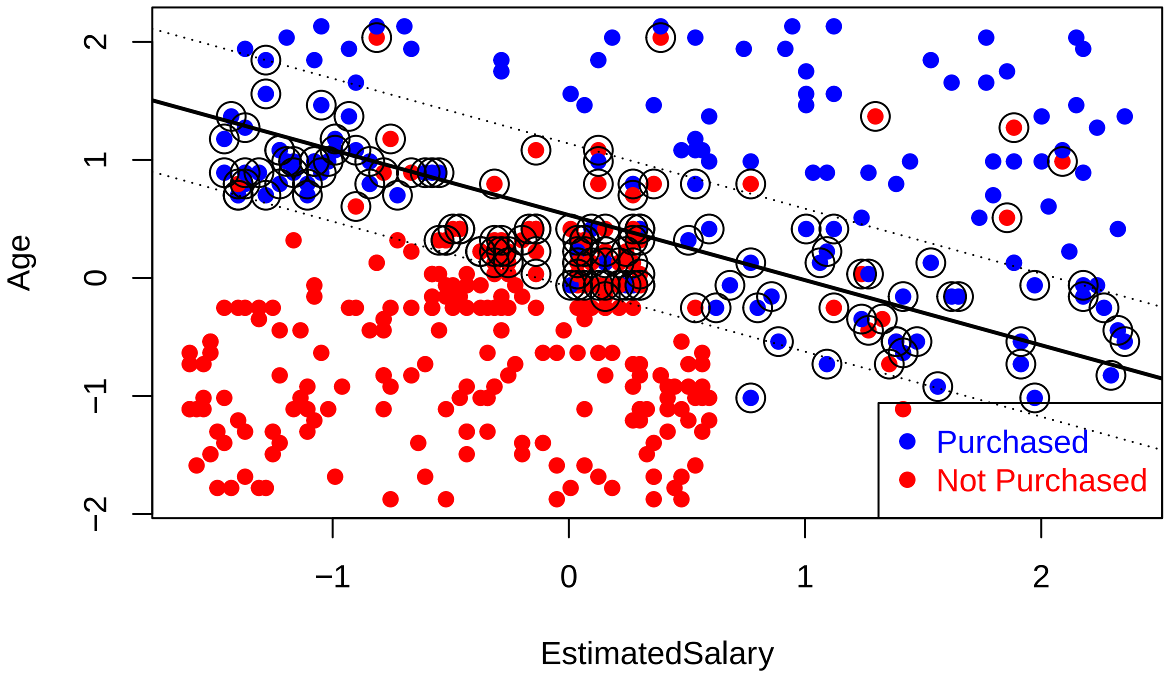
```
b <- t(svm.fit$coefs) %*% svm.fit$SV
b0 <- -svm.fit$rho
x = scaled_vars

# plot
plot(
  scaled_vars,
  col = ifelse(use_data$Purchased > 0, "blue", "red"),
  pch = 19,
  xlab = "EstimatedSalary",
  ylab = "Age"
)
legend(
  "bottomright",
  c("Purchased", "Not Purchased"),
  col = c("blue", "red"),
  pch = c(19, 19),
  text.col = c("blue", "red")
)
# Decision line
abline(
  a = -b0 / b[1, 2],
  b = -b[1, 1] / b[1, 2],
  col = "black",
  lty = 1,
  lwd = 2
)

# Marking the support vectors
points(x[svm.fit$index, ], col = "black", cex = 2)    # using cex='2'

# Two margin lines
abline(
  a = (-b0 - 1) / b[1, 2],
```

```
  b = -b[1, 1] / b[1, 2],
  col = "black",
  lty = 3
)
abline(
  a = (-b0 + 1) / b[1, 2],
  b = -b[1, 1] / b[1, 2],
  col = "black",
  lty = 3
)
```



The plot shows the decision line with marked support vectors.

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.

```
# Loading the libraries
library(kernlab)
library(caret)
```

```
## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:kernlab':
```

```
##
##     alpha
```

```
## Loading required package: lattice
```

```
library(ElemStatLearn)
```

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

```
# subset the data to include only two digits: 4 and 9
train = data.frame(zip.train)
train = subset(train, X1 == 9 | X1 == 4)

# subset the data to include only two digits: 4 and 9
test = data.frame(zip.test)
test = subset(test, X1 == 9 | X1 == 4)
```

- [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.

```
# Constructing grid for 'C' parameter
cost.grid = expand.grid(C = seq(0.01, 2, length = 20))
# 10-fold cross-validation
train_control = trainControl(method = "cv", number = 10)

svm2 <-
  train(
    as.factor(X1) ~ .,
    data = train,
    method = "svmLinear",
    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: 1167, 1165, 1166, 1166, 1166, 1167, ...
## Resampling results across tuning parameters:
##
##   C          Accuracy   Kappa
##   0.0100000  0.9884019  0.9768010
##   0.1147368  0.9845558  0.9691095
```

```
##   0.2194737  0.9837806  0.9675599
##   0.3242105  0.9830113  0.9660215
##   0.4289474  0.9837865  0.9675729
##   0.5336842  0.9837865  0.9675729
##   0.6384211  0.9837865  0.9675729
##   0.7431579  0.9837865  0.9675729
##   0.8478947  0.9837865  0.9675729
##   0.9526316  0.9837865  0.9675729
##   1.0573684  0.9837865  0.9675729
##   1.1621053  0.9837865  0.9675729
##   1.2668421  0.9837865  0.9675729
##   1.3715789  0.9837865  0.9675729
##   1.4763158  0.9837865  0.9675729
##   1.5810526  0.9837865  0.9675729
##   1.6857895  0.9837865  0.9675729
##   1.7905263  0.9837865  0.9675729
##   1.8952632  0.9837865  0.9675729
##   2.0000000  0.9837865  0.9675729
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.01.
```

Accuracy was used to select the optimal model using the largest value. The final value used for the model was C = 0.01.

```
# Making test data predictions using best model obtained from above
linear_svm_Y.pred = predict(svm2, test[, -1])

# Confusion table of results
confusion_table_2 = table(test[, 1], linear_svm_Y.pred)
confusion_table_2
```

```
##    linear_svm_Y.pred
##       4   9
##   4 192   8
##   9   5 172
```

```
100 * (confusion_table_2[1, 1] + confusion_table_2[2, 2]) / length(linear_svm_Y.pred)
```

```
## [1] 96.55172
```

The testing data accuracy using the SVM model with C = 0.01 is 96.55172%.

- [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.

```
# Tuning parameters using 10-fold CV for radial Kernel SVM
svm.radial <- train(
```

```r
  as.factor(X1) ~ .,
  data = train,
  method = "svmRadial",
  preProcess = c("center", "scale"),
  tuneGrid = expand.grid(C = c(0.01, 0.1, 0.5, 1), sigma = c(0.01, 0.02)),
  trControl = trainControl(method = "cv", number = 10)
)
```

```r
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: 1167, 1166, 1167, 1166, 1166, 1166, ...
## Resampling results across tuning parameters:
##
##   C     sigma  Accuracy   Kappa
##   0.01  0.01   0.5324091  0.05931957
##   0.01  0.02   0.5030888  0.00000000
##   0.10  0.01   0.9074240  0.81456385
##   0.10  0.02   0.8047764  0.60856451
##   0.50  0.01   0.9521646  0.90425970
##   0.50  0.02   0.9051103  0.80992751
##   1.00  0.01   0.9652713  0.93050100
##   1.00  0.02   0.9297973  0.85943379
##
## 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 = 1.
```

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 = 1.

```r
# Making test data predictions using the best radial kernel SVM obtained froma above
radial_svm_Y.pred = predict(svm.radial, test[, -1])

# Confusion table of results
confusion_table_3 = table(test[, 1], radial_svm_Y.pred)
confusion_table_3
```

```
##    radial_svm_Y.pred
##       4   9
##   4 197   3
##   9  12 165
```

```r
100 * (confusion_table_3[1, 1] + confusion_table_3[2, 2]) / length(radial_svm_Y.pred)
```

```
## [1] 96.02122
```

The testing data accuracy using the best obtained radial kernel SVM is 96.02122%.