Assignment 03

Chandima Attanayake

2025-04-26

**Part A**

*QA1. Difference between SVM with hard margin and soft margin*

Hard Margin SVM assumes that the data is perfectly separable — it tries to find a hyperplane that separates the data without any errors (no points are allowed inside the margin).

Soft Margin SVM allows for some misclassifications as it introduces slack variables to tolerate violations, aiming for better generalization to unseen data.

*QA2. Role of the cost parameter (C) in Soft Margin Classifier(SVM)*

The C parameter controls the trade-off between achieving a low training error and a low testing error.

A high C tries to classify all training examples correctly (less tolerance for errors, less regularization).

A low C allows more misclassifications but improves the generalization (more regularization).

*QA3. Perceptron Activation*

Calulcation of Activation = (x1 × w1) + (x2 × w2) + bias =(0.1×0.8)+(11.1×−0.2)+2.8 = 0.66

The perceptron activation output is 0.66.

*QA4. Role of alpha (learning rate) in the delta rule*

The learning rate alpha determines how much the weights are adjusted during each step of learning.

A small alpha leads to slow but stable convergence.

A large alpha can make learning faster but risks overshooting the optimal solution.

**Part B**

# Load Libraries  
library(ISLR)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1-8

library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

# Select Required Variables  
Carseats\_Filtered <- Carseats %>%  
 select(Sales, Price, Advertising, Population, Age, Income, Education)

*QB1. Build Linear SVM Regression Model*

# Set seed for reproducibility  
set.seed(123)  
  
# Train Linear SVM model  
svm\_model <- train(Sales ~ ., data = Carseats\_Filtered,  
 method = "svmLinear",  
 trControl = trainControl(method = "cv", number = 5))  
  
# Check model results  
print(svm\_model)

## Support Vector Machines with Linear Kernel   
##   
## 400 samples  
## 6 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 320, 321, 319, 320, 320   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 2.305688 0.3442665 1.85399  
##   
## Tuning parameter 'C' was held constant at a value of 1

# Get R-squared  
svm\_model$results$Rsquared

## [1] 0.3442665

*QB2. Customize the Search Grid for Different C Values*

# Set custom tuning grid  
grid <- expand.grid(C = c(0.1, 0.5, 1, 10))  
  
# Train the model with customized grid  
svm\_model\_grid <- train(Sales ~ ., data = Carseats\_Filtered,  
 method = "svmLinear",  
 tuneGrid = grid,  
 trControl = trainControl(method = "repeatedcv",  
 number = 5, repeats = 2))  
  
# Check model results  
print(svm\_model\_grid)

## Support Vector Machines with Linear Kernel   
##   
## 400 samples  
## 6 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold, repeated 2 times)   
## Summary of sample sizes: 320, 320, 320, 319, 321, 320, ...   
## Resampling results across tuning parameters:  
##   
## C RMSE Rsquared MAE   
## 0.1 2.289763 0.3486459 1.844170  
## 0.5 2.288391 0.3497639 1.842448  
## 1.0 2.288953 0.3496537 1.842709  
## 10.0 2.288642 0.3497177 1.842417  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was C = 0.5.

*QB3. Train a Neural Network Model*

# Preprocess (Scale and Center)  
preproc <- preProcess(Carseats\_Filtered, method = c("center", "scale"))  
Carseats\_Scaled <- predict(preproc, Carseats\_Filtered)  
  
# Train Neural Network  
set.seed(123)  
nn\_model <- train(Sales ~ ., data = Carseats\_Scaled,  
 method = "nnet",  
 trControl = trainControl(method = "cv", number = 5),  
 linout = TRUE, trace = FALSE)  
  
# Check model results  
print(nn\_model)

## Neural Network   
##   
## 400 samples  
## 6 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 320, 321, 320, 319, 320   
## Resampling results across tuning parameters:  
##   
## size decay RMSE Rsquared MAE   
## 1 0e+00 0.8078265 0.3509576 0.6427277  
## 1 1e-04 0.8078214 0.3509693 0.6427205  
## 1 1e-01 0.8051115 0.3552040 0.6398902  
## 3 0e+00 0.8796669 0.2497132 0.7020197  
## 3 1e-04 0.8809579 0.2752107 0.6935927  
## 3 1e-01 0.8481082 0.3014053 0.6868067  
## 5 0e+00 0.9094078 0.2609347 0.7308954  
## 5 1e-04 0.9034134 0.2426361 0.7243657  
## 5 1e-01 0.8501525 0.3060133 0.6819181  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were size = 1 and decay = 0.1.

# Get R-squared  
nn\_model$results$Rsquared

## [1] 0.3509576 0.3509693 0.3552040 0.2497132 0.2752107 0.3014053 0.2609347  
## [8] 0.2426361 0.3060133

*QB4. Predict Sales for New Input Using Neural Net Model*

# New input data  
new\_data <- data.frame(  
 Sales = 9,  
 Price = 6.54,  
 Advertising = 0,  
 Population = 124,  
 Age = 76,  
 Income = 110,  
 Education = 10  
)  
  
# Scale new data  
new\_data\_scaled <- predict(preproc, new\_data)  
  
# Drop 'Sales' before prediction  
new\_data\_scaled <- new\_data\_scaled %>% select(-Sales)  
  
# Predict Sales using the trained neural network model  
predicted\_sales <- predict(nn\_model, newdata = new\_data\_scaled)  
  
# Output predicted sales  
predicted\_sales

## 1   
## 1.398309