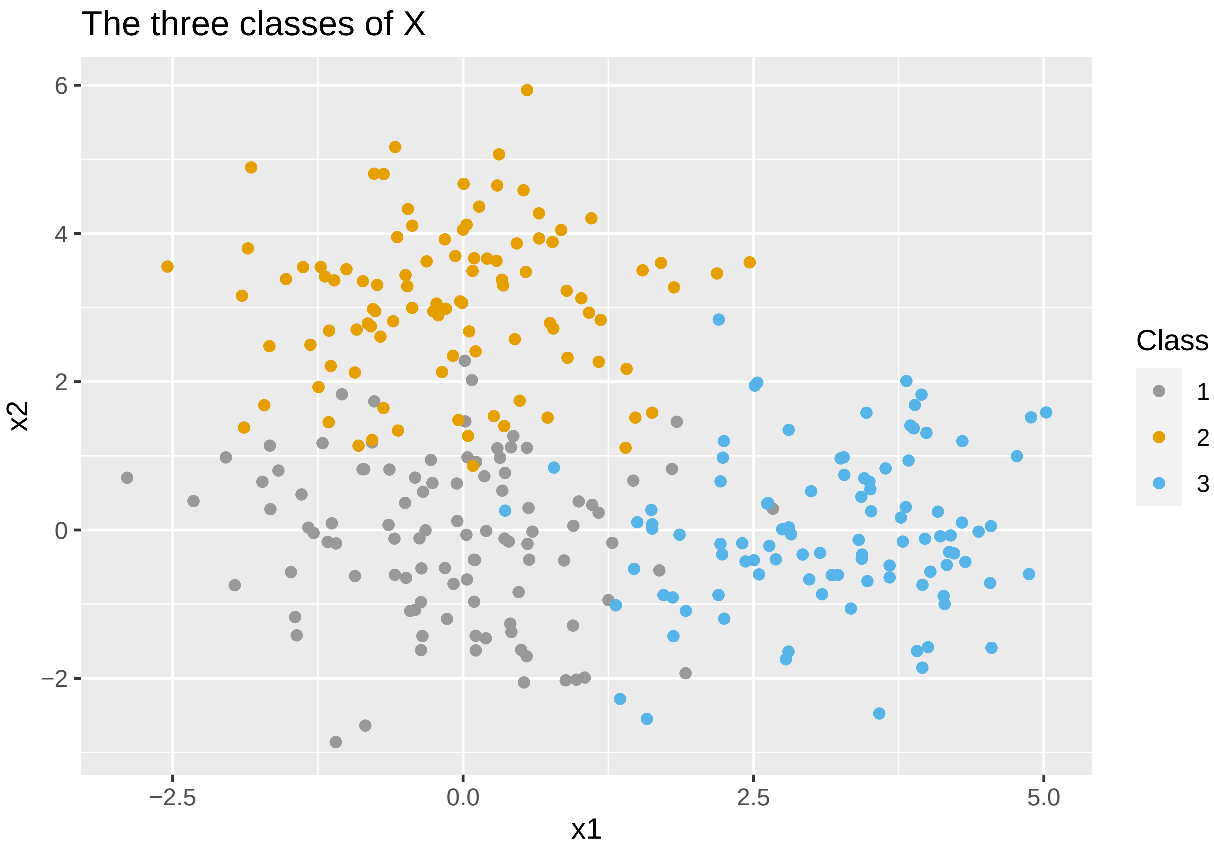
MAST90083 Assignment 3

Haonan Zhong

Question 1.1

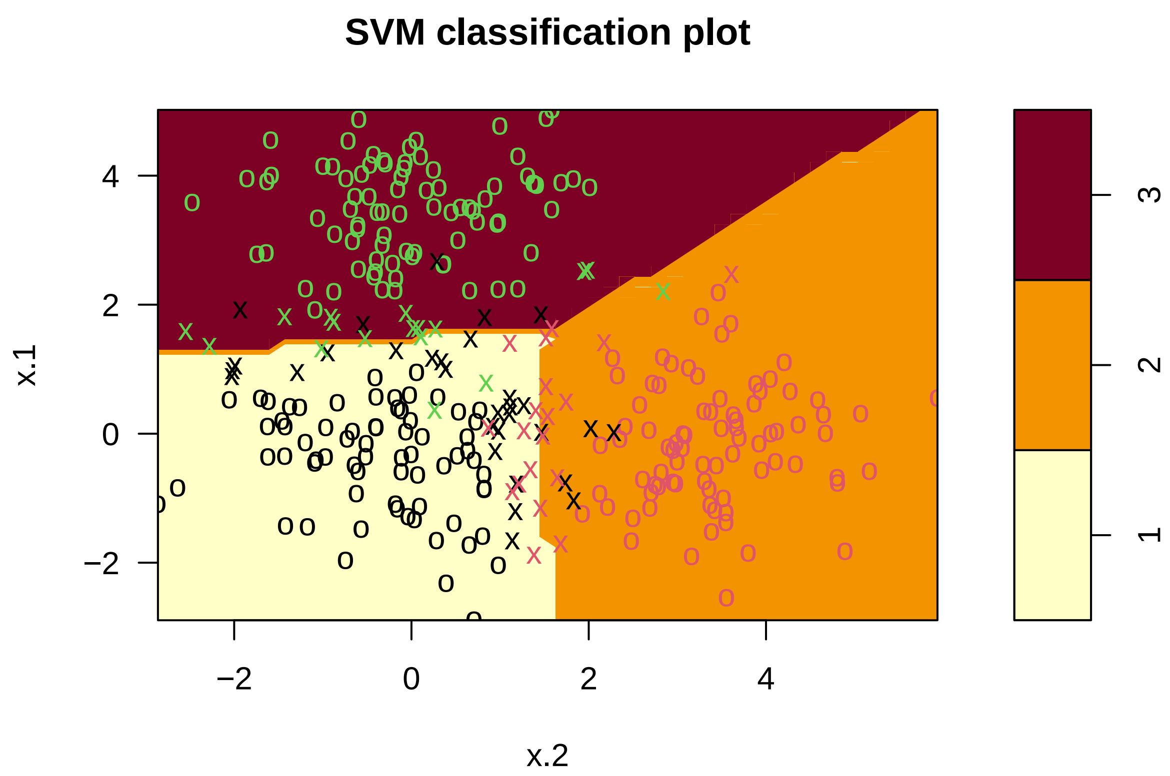
```
library(ggplot2)
library(dplyr)
library(e1071)

C <- 3
N <- 300

set.seed(50)
# Construct x, y, and z
x <- matrix(rnorm(N*2), ncol = 2)
z <- matrix(c(0, 0, 3, 0, 3, 0), C, 2)
y <- c(rep(1, 100), rep(2, 100), rep(3, 100))

# Assign class specific means to data points
px <- pnorm(x)
for (i in 1:N) {
  p.1 <- px[i,1]
  p.2 <- px[i,2]
  if (y[i] == 1) {
    x[i,1] <- qnorm(p.1, z[1,1])
    x[i,2] <- qnorm(p.2, z[1,2])
  }
  else if (y[i] == 2) {
    x[i,1] <- qnorm(p.1, z[2,1])
    x[i,2] <- qnorm(p.2, z[2,2])
  }
  else {
    x[i,1] <- qnorm(p.1, z[3,1])
    x[i,2] <- qnorm(p.2, z[3,2])
  }
}
```

```
x_df <- data.frame(x)
colnames(x_df) <- c('x1', 'x2')

x_df %>%
  ggplot(aes(x = x1, y = x2), xlab = "x1", ylab = "x2") +
  geom_point(aes(color = as.factor(y))) +
  ggtitle("The three classes of X") +
  scale_color_manual(values=c("#999999", "#E69F00", "#56B4E9")) +
  labs(color='Class', x = "x1", y = "x2")
```



Question 1.2

```
tdata <- data.frame(x = x, y = as.factor(y))
svmfit <- svm(y ~., data = tdata, cost = 10, kernel = "linear")
plot(svmfit, tdata)
```




```r
# Generate summary using the object svmfit
summary(svmfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = tdata, cost = 10, kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##
## Number of Support Vectors:  67
##
##  ( 31 19 17 )
##
##
## Number of Classes:  3
##
## Levels:
##  1 2 3
```

As we can see from the summary output, there are a total of 67 support vectors, where 31 are from class 1, following by 19 from class 2, while 17 are from class 3.

Question 1.3

```r
set.seed(50)
tuned <- tune(svm, y~., data = tdata, kernel = "linear",
              ranges=list(cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
summary(tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.08
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-03 0.77000000 0.04288946
## 2 1e-02 0.10000000 0.04969040
## 3 1e-01 0.10666667 0.07503086
## 4 1e+00 0.08000000 0.05921294
## 5 5e+00 0.09000000 0.05889937
## 6 1e+01 0.09000000 0.05889937
## 7 1e+02 0.08333333 0.05270463
```

As we can see from the result above, the minimum cross validation error is 0.08 when the $cost = 1$.

```r
bestmod <- tuned$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = y ~ ., data = tdata, ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5, 10, 100)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  81
##
##  ( 37 22 22 )
##
##
## Number of Classes:  3
##
## Levels:
##  1 2 3
```

As the result shown, the number of support vector has increased from 67 to 81. 37 of the support vectors are from class 1, and both class 2 and class 3 has 22 support vectors.

Question 1.4

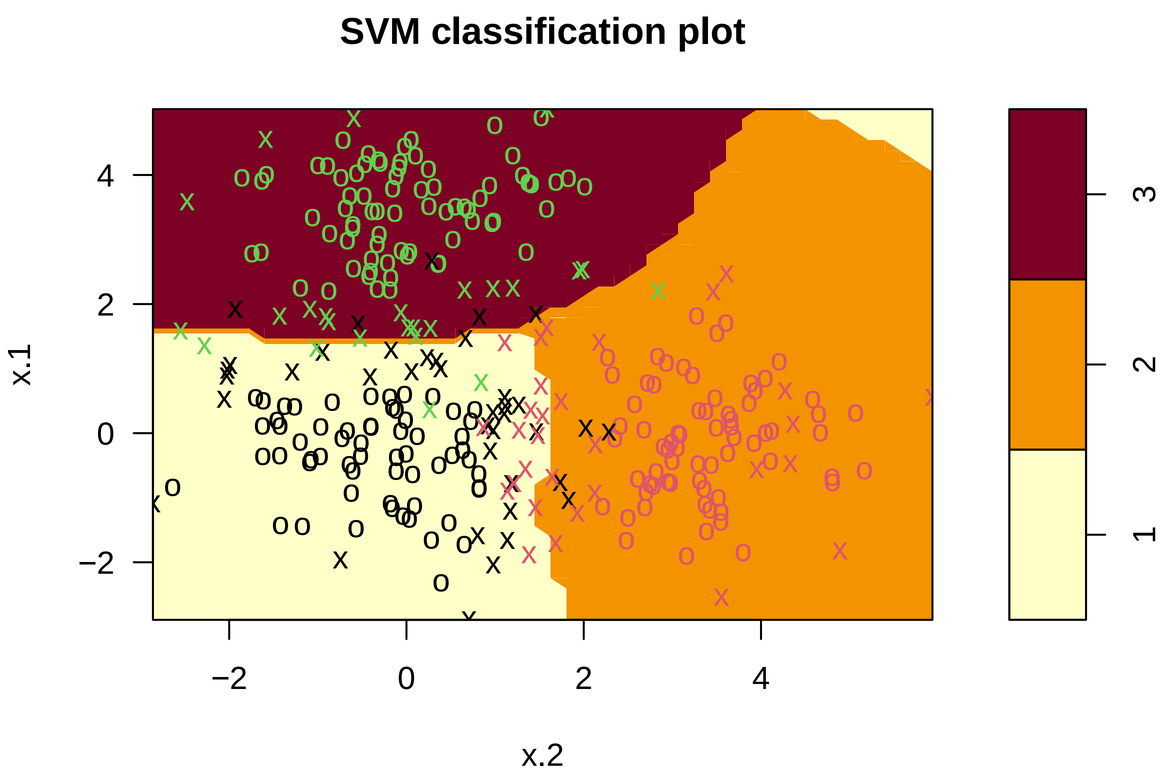
```
set.seed(10)
x_test <- matrix(rnorm(N*2), ncol=2)

set.seed(100)
y_test <- sample(c(1:3), 300, replace = TRUE)

# Assign class specific means to data points
px <- pnorm(x_test)
for (i in 1:N) {
  p.1 <- px[i,1]
  p.2 <- px[i,2]
  if (y_test[i] == 1) {
    x_test[i,1] <- qnorm(p.1, z[1,1])
    x_test[i,2] <- qnorm(p.2, z[1,2])
  }
  else if (y_test[i] == 2) {
    x_test[i,1] <- qnorm(p.1, z[2,1])
    x_test[i,2] <- qnorm(p.2, z[2,2])
  }
  else {
    x_test[i,1] <- qnorm(p.1, z[3,1])
    x_test[i,2] <- qnorm(p.2, z[3,2])
  }
}

testdata <- data.frame(x = x_test , y = as.factor(y_test))

set.seed(100)
y_p <- predict(bestmod, testdata)
# Print the results in form of a table for the predicted labels against the test labels
table(y_p, y_test)
```

```
##    y_test
## y_p   1   2   3
##   1  72   7   5
##   2  10  92   0
##   3   4   1 109
```

As shown in the confusion matrix above, we can see there are 27 misclassified observations. And we can see there are 109 correctly classified observations for class 3. It is reasonable given that y is labeled randomly, as we can see there are a total number of 114 observation labeled as class 3.

Question 1.5

```r
svmfit_radial <- svm(y ~., data = tdata, cost = 1, gamma = 1, kernel = "radial")
plot(svmfit_radial, tdata)
```




```r
# Generate summary using the object svmfit with radial kernel
summary(svmfit_radial)
```

```
##
## Call:
## svm(formula = y ~ ., data = tdata, cost = 1, gamma = 1, kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  94
##
##  ( 39 30 25 )
##
##
## Number of Classes:  3
##
## Levels:
##  1 2 3
```

As we can see the summary output of the SVM with radial kernel, there are a total of 94 support vectors. Where class 1, class 2, and class 3 have 39, 30, and 25, respectively.

```r
set.seed(50)
# Tuning, perform a ten-fold cross-validation
svmfit_radial_cv <- tune(svm, y ~., data = tdata, kernel = "radial",
                         ranges = list(cost = c(0.1, 1, 10, 100, 1000),
                                       gamma = c(0.5, 1, 2, 3, 4)))

summary(svmfit_radial_cv)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##   100     2
##
## - best performance: 0.07333333
##
## - Detailed performance results:
##     cost gamma      error dispersion
## 1  1e-01   0.5 0.10000000 0.06478835
## 2  1e+00   0.5 0.08666667 0.06324555
## 3  1e+01   0.5 0.07666667 0.05454639
## 4  1e+02   0.5 0.08000000 0.04766136
## 5  1e+03   0.5 0.09000000 0.04981447
## 6  1e-01   1.0 0.10666667 0.07503086
## 7  1e+00   1.0 0.09000000 0.06295207
## 8  1e+01   1.0 0.08000000 0.04216370
## 9  1e+02   1.0 0.08000000 0.03583226
## 10 1e+03   1.0 0.07666667 0.05223404
## 11 1e-01   2.0 0.09333333 0.06440612
## 12 1e+00   2.0 0.08333333 0.05719795
## 13 1e+01   2.0 0.07666667 0.04727122
## 14 1e+02   2.0 0.07333333 0.04097575
## 15 1e+03   2.0 0.10333333 0.05973191
## 16 1e-01   3.0 0.09000000 0.06295207
## 17 1e+00   3.0 0.07666667 0.05454639
## 18 1e+01   3.0 0.07666667 0.04458312
## 19 1e+02   3.0 0.08666667 0.04766136
## 20 1e+03   3.0 0.12000000 0.06126244
## 21 1e-01   4.0 0.09333333 0.06813204
## 22 1e+00   4.0 0.08000000 0.04766136
## 23 1e+01   4.0 0.07333333 0.04388537
## 24 1e+02   4.0 0.10333333 0.05544433
## 25 1e+03   4.0 0.10666667 0.06992059
```

As we can see from the summary above, the minimum cross validation error is 0.07333333 when $gamma = 2$ and $cost = 100$.

```r
# Printing the summary of the best model with radial kernel
bestmod_radial <- svm(y ~., data = tdata, cost = 100, gamma = 2, kernel = "radial")
summary(bestmod_radial)
```

```
##
## Call:
## svm(formula = y ~ ., data = tdata, cost = 100, gamma = 2, kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  100
##
## Number of Support Vectors:  88
##
##  ( 32 30 26 )
##
##
## Number of Classes:  3
##
## Levels:
##  1 2 3
```

As we can see from the summary above, the number of support vectors decreases from 94 to 88 compared to the previous model. And class 1, class 2, and class 3 have 32, 30, and 26 support vectors, respectively.

```r
set.seed(100)
y_p <- predict(bestmod_radial, testdata)
# Print the results in form of a table for the predicted labels against the test labels
table(y_p, y_test)
```

```
##    y_test
## y_p   1   2   3
##   1  67   4   9
##   2  15  96   3
##   3   4   0 102
```

As the confusion matrix shown above, there are 35 observation being misclassified, which is more than the SVM fitted using linear kernel. Thus, given the data points are linearly separable and we don't need the radial kernel.

Question 2.1

The hidden node can be express as follow, where f_j is the activation function of the hidden layer

$$Z_j = f_j(\beta_{0j} + \sum_{m=1}^{r} \beta_{mj} \cdot x_m)$$

The output node Y_k can be express as follow, where g_k is the activation function for the output layer

$$Y_k = g_k[\alpha_{0k} + \sum_{j=1}^{t} \alpha_{jk} \cdot f_j(\beta_{0j} + \sum_{m=1}^{r} \beta_{mj} \cdot x_m)]$$

Question 2.2

The hidden layer can be express as follow using matrix form, where f is the activation function of the hidden layer, $\mathbf{Z} = (Z_1, ..., Z_t)^\top$ is a vector of the hidden nodes, $\beta_\mathbf{0} = (\beta_{01}, ..., \beta_{0t})^\top$ is a vector of biases of the hidden nodes, $\mathbf{B} = (\beta_1, ..., \beta_t)^\top$ is a matrix with $(t \times r)$ rows of connection weight for each hidden nodes, and $\mathbf{X} = (X_1, ..., X_r)^\top$ is matrix of the input.

$$\mathbf{Z} = f(\beta_\mathbf{0} + \mathbf{X}^\top \mathbf{B})$$

The output layer can be express as follow using matrix form, where g is the activation function of the hidden layer, $\mathbf{Y} = (Y_1, ..., Y_t)^\top$ is a vector of the output nodes, $\alpha_\mathbf{0} = (\alpha_{01}, ..., \alpha_{0s})^\top$ is a vector of biases of the hidden nodes, $\alpha = (\alpha_1, ..., \alpha_s)^\top$ is a matrix with $(s \times t)$ rows of connection weight for each output nodes, and $\mathbf{Z}$ is described in the above function.

$$\mathbf{Y} = g(\alpha_\mathbf{0} + \alpha \mathbf{Z})$$

Question 2.3

If the activation function $f_j()$ fo the hidden layer is simply the identity function, and the actiavtion function $g_k()$ is the same as the one of the single-layer perceptron, then the two-layer network will be equivalent to a single layer perceptron.

Question 2.4

If both activation functions for the hidden layer and output layer are taken to be identity functions, then the network simply be a linear combination of the input values, which is essentially a linear regression model.