MATH 3190 Final Project

Jun Hanvey, Ian McFarlane, Kellen Nankervis

Due April 25, 2024

Hello World!

```r
print("Hello world!")
```

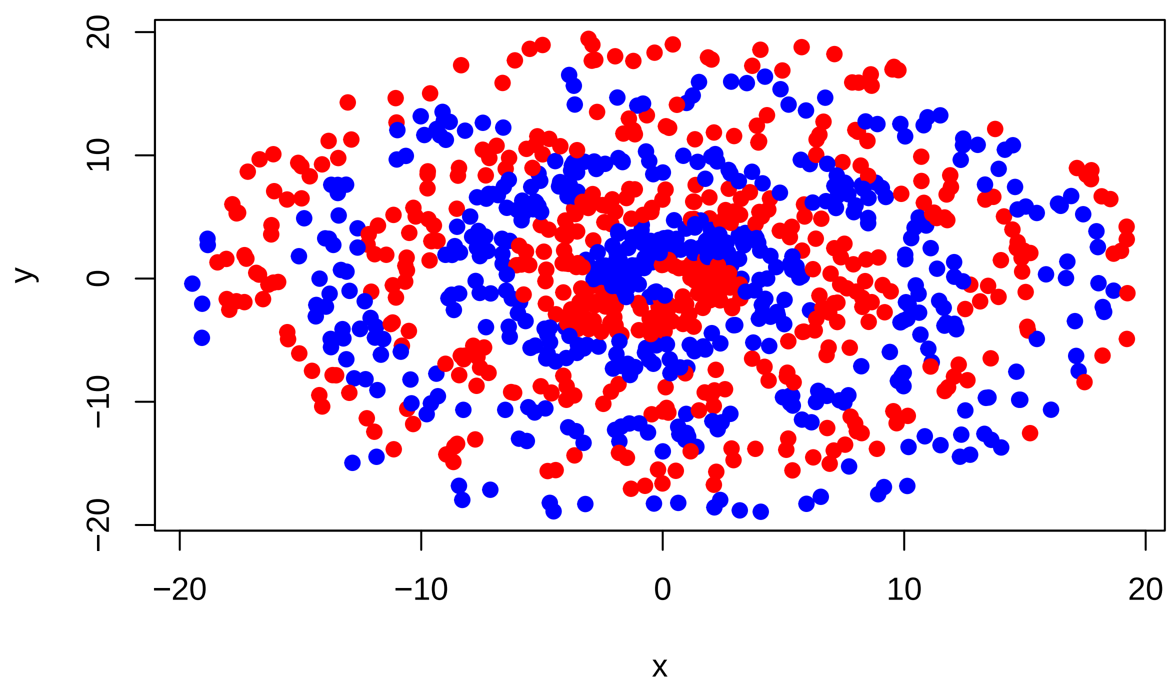
```
## [1] "Hello world!"
```

#TODO: Find good value for n and fix colors so they are the same as the later plot. - Kellen Nankervis
Create example data to test RBF kernel function

```r
set.seed(123)
n <- 1000
for (i in 1:10) {
  r <- runif(n, 1, 6 * pi + 1)
  theta <- runif(n, 0, 2 * pi)
}
# Make a sample classification n observations long
class <- sample(c(0, 1), n, replace = TRUE)
# Now fill the classification vector with the correct values
for (j in 1:n) {
  if ((r[j] + theta[j]) %% (2 * pi) < pi) {
    class[j] <- 1
  } else {
    class[j] <- 0
  }
}

# Create a data frame with the data
data <- data.frame(r, theta, class)
# Create a scatter plot of the data in x and y coordinates
data$color <- ifelse(data$class == 1, "red", "blue")
data$x <- data$r * cos(data$theta)
data$y <- data$r * sin(data$theta)
```

#TODO: Decide which plots to show and which to delete. Commented out ones would be my current suggestions to delete or at least move to later in the document. Make plots look good when knitted to pdf and/or slides. - Kellen Nankervis

```r
# Plot the data in the x and y coordinates
plot(data$x, data$y, col = data$color, pch = 19, xlab = "x", ylab = "y")
```



```
# Plot the data in polar coordinates
# plot(data$r, data$theta, col = data$color, pch = 19, xlab = "r", ylab = "theta")

# Plot r*theta vs. r^2 * theta^2
# plot(data$r + data$theta, data$r * data$theta, col = data$color, pch = 19, xlab = "r*theta", ylab = "
```

#TODO: Decide on good cost to not overfit. Could be a good spot to also show how we can use cross-validation to find the best cost. Make plots look good when knitted to pdf and/or slides. - Kellen Nankervis

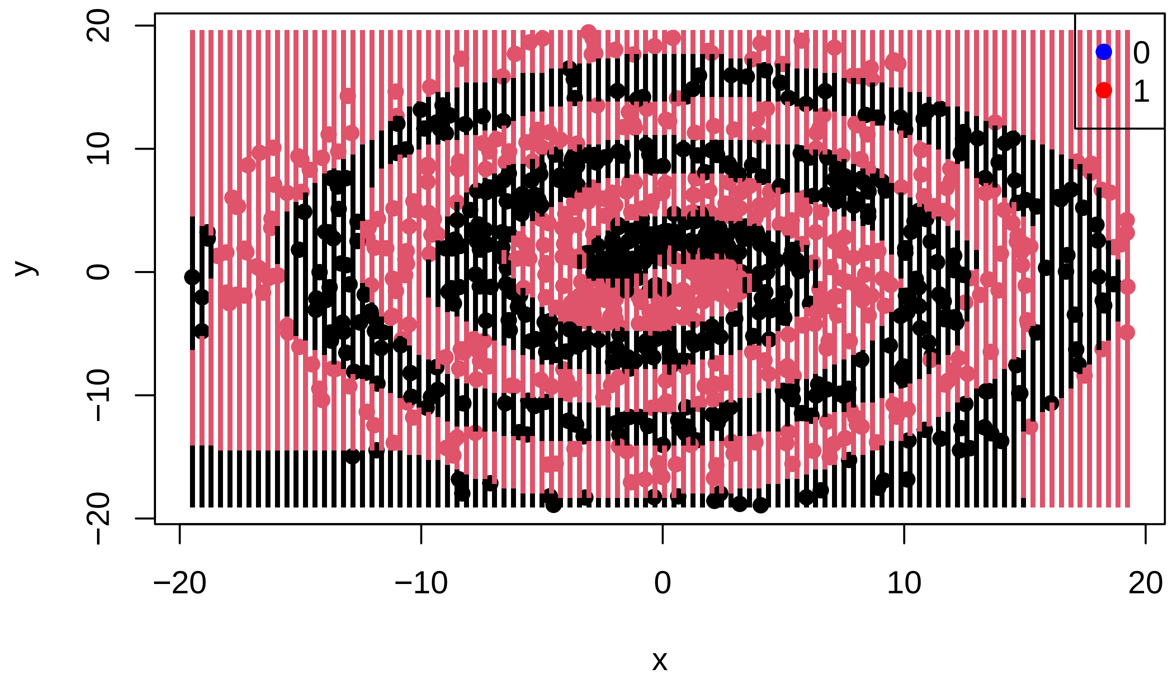
```
# Load the required svm library
library(e1071)
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
# Convert class to a factor
data$class <- as.factor(data$class)

# Create a data frame with only the class and the x and y coordinates
data2 <- data.frame(class = data$class, x = data$x, y = data$y)
data2$class <- as.factor(data2$class)

# Use a radial basis function kernel to classify the data with the SVM function
svmfit <- svm(class ~ ., data = data2, kernel = "radial", cost = 100000, gamma = 1)

print(svmfit)
```

```
##
## Call:
## svm(formula = class ~ ., data = data2, kernel = "radial", cost = 1e+05,
##     gamma = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1e+05
##
## Number of Support Vectors:  349
```

```
# Plot the data points
plot(data2$x, data2$y, col = data2$class, pch = 19, xlab = "x", ylab = "y")

# Plot the decision boundary
# plot(svmfit, data2$class, grid = 100, dataSymbol = 16)
x1_grid <- seq(min(data2$x), max(data2$x), length.out = 100)
x2_grid <- seq(min(data2$y), max(data2$y), length.out = 100)
grid <- expand.grid(x = x1_grid, y = x2_grid)

predicted_labels <- predict(svmfit, newdata = grid)

plot(data2$x, data2$y, col = data2$class, pch = 19, xlab = "x", ylab = "y")
points(grid$x, grid$y, col = factor(predicted_labels), pch = ".", cex = 2.5)
legend("topright", legend = levels(data2$class), col = c("blue", "red"), pch = 19)
```



```r
# Create a confusion matrix to evaluate the SVM model
confusionMatrix(predict(svmfit, data2), data2$class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 484  21
##          1  23 472
##
##                Accuracy : 0.956
##                  95% CI : (0.9414, 0.9679)
##     No Information Rate : 0.507
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.912
##
##  Mcnemar's Test P-Value : 0.8802
##
##             Sensitivity : 0.9546
##             Specificity : 0.9574
##          Pos Pred Value : 0.9584
##          Neg Pred Value : 0.9535
##              Prevalence : 0.5070
##          Detection Rate : 0.4840
##    Detection Prevalence : 0.5050
##       Balanced Accuracy : 0.9560
##
##        'Positive' Class : 0
##
```

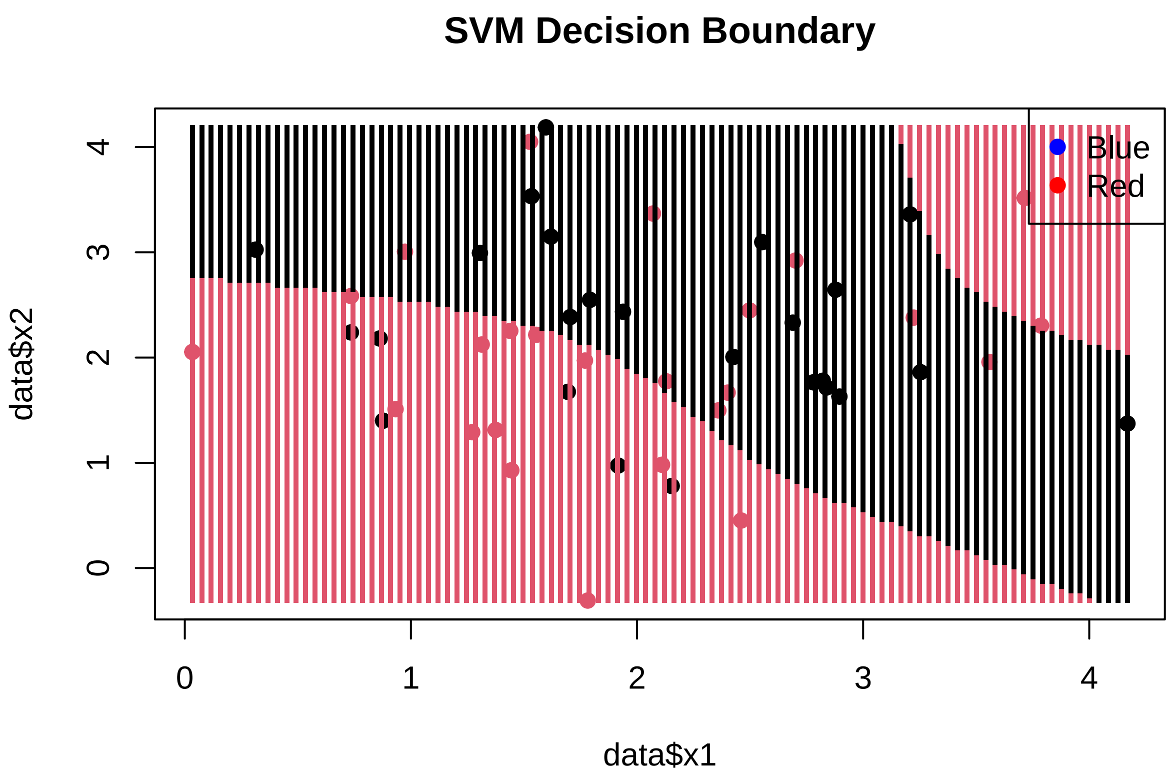
#This was some example code I found to make the decision boundary plot. I'm leaving it for others to see but it will be removed in the final version obviously. - Kellen Nankervis

```r
print("")
```

```
## [1] ""
```

```r
# Create a toy dataset
set.seed(123)
data <- data.frame(
  x1 = rnorm(50, mean = 2),
  x2 = rnorm(50, mean = 2),
  label = c(rep("Red", 25), rep("Blue", 25)) |> as.factor()
)

# Train an SVM
svm_model <- svm(label ~ ., data = data, kernel = "radial")

# Create a grid of points for prediction
x1_grid <- seq(min(data$x1), max(data$x1), length.out = 100)
x2_grid <- seq(min(data$x2), max(data$x2), length.out = 100)
grid <- expand.grid(x1 = x1_grid, x2 = x2_grid)

# Predict class labels for the grid
```

```r
predicted_labels <- predict(svm_model, newdata = grid)

# Plot the decision boundary
plot(data$x1, data$x2, col = factor(data$label), pch = 19, main = "SVM Decision Boundary")
points(grid$x1, grid$x2, col = factor(predicted_labels), pch = ".", cex = 2.5)
legend("topright", legend = levels(data$label), col = c("blue", "red"), pch = 19)
```


SVM Decision Boundary
data$x2
data$x1
Blue
Red