RBF Kernels: Generating a complex dataset

SUPPORT VECTOR MACHINES IN R

Kailash Awati
Instructor

A bit about RBF Kernels

- Highly flexible kernel.
  - Can fit complex decision boundaries.
- Commonly used in practice.

Generate a complex dataset

- 600 points (x1, x2)
- x1 and x2 distributed differently

```
n <- 600
set.seed(42)

df <- data.frame(x1 = rnorm(n, mean = -0.5, sd = 1),
                 x2 = runif(n, min = -1, max = 1))
```

Generate boundary

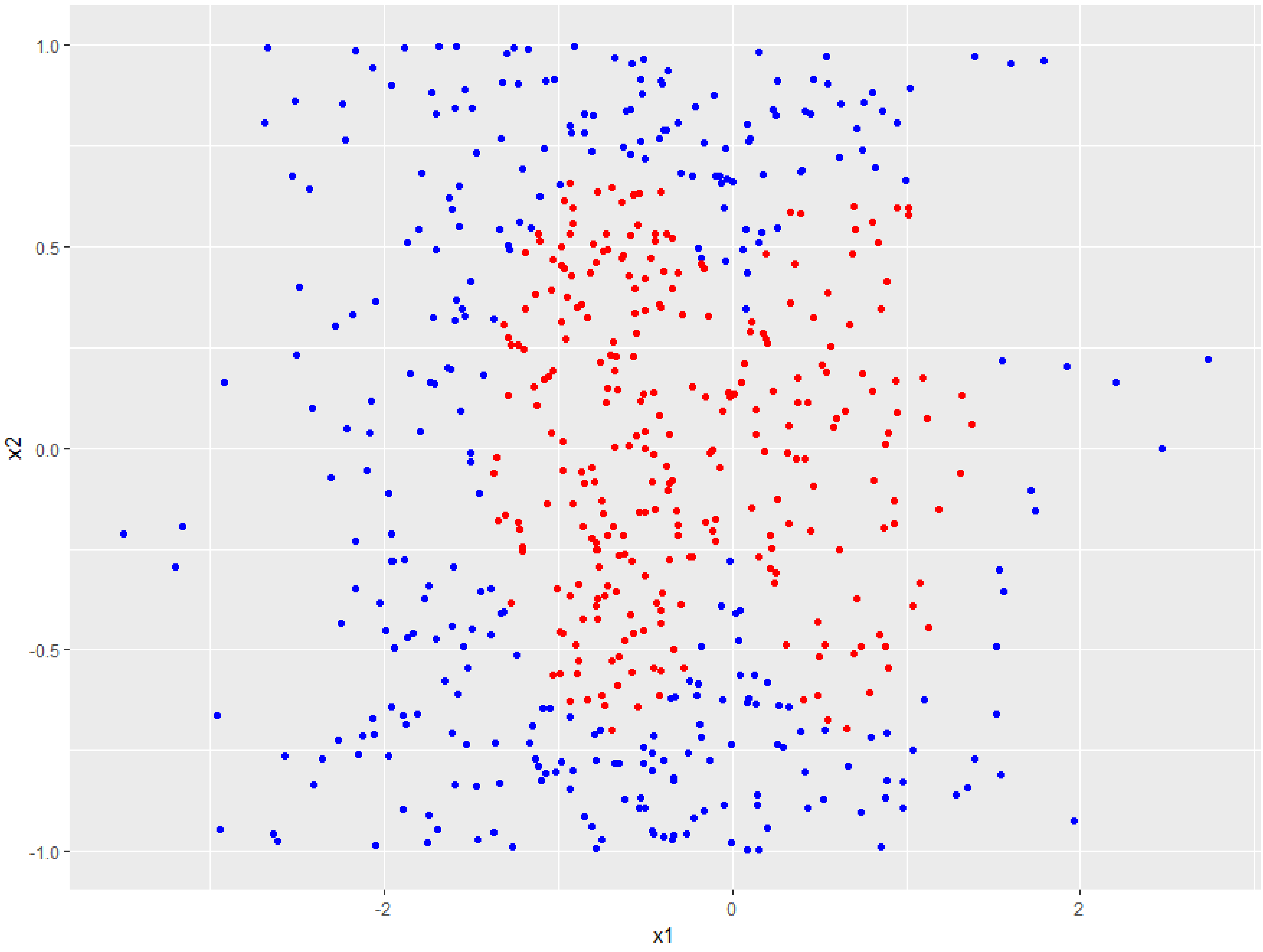
- Boundary consists of two equi-radial circles with a single point in common.

```
# Set radius and centers
radius <- 0.7
radius_squared <- radius ^ 2
center_1 <- c(-0.7, 0)
center_2 <- c(0.7, 0)
# Classify points
df$y <-
    factor(ifelse(
        (df$x1 - center_1[1]) ^ 2 + (df$x2 - center_1[2]) ^ 2 < radius_squared |
        (df$x1 - center_2[1]) ^ 2 + (df$x2 - center_2[2]) ^ 2 < radius_squared,
        -1, 1), levels = c(-1, 1))
```

Visualizing the dataset

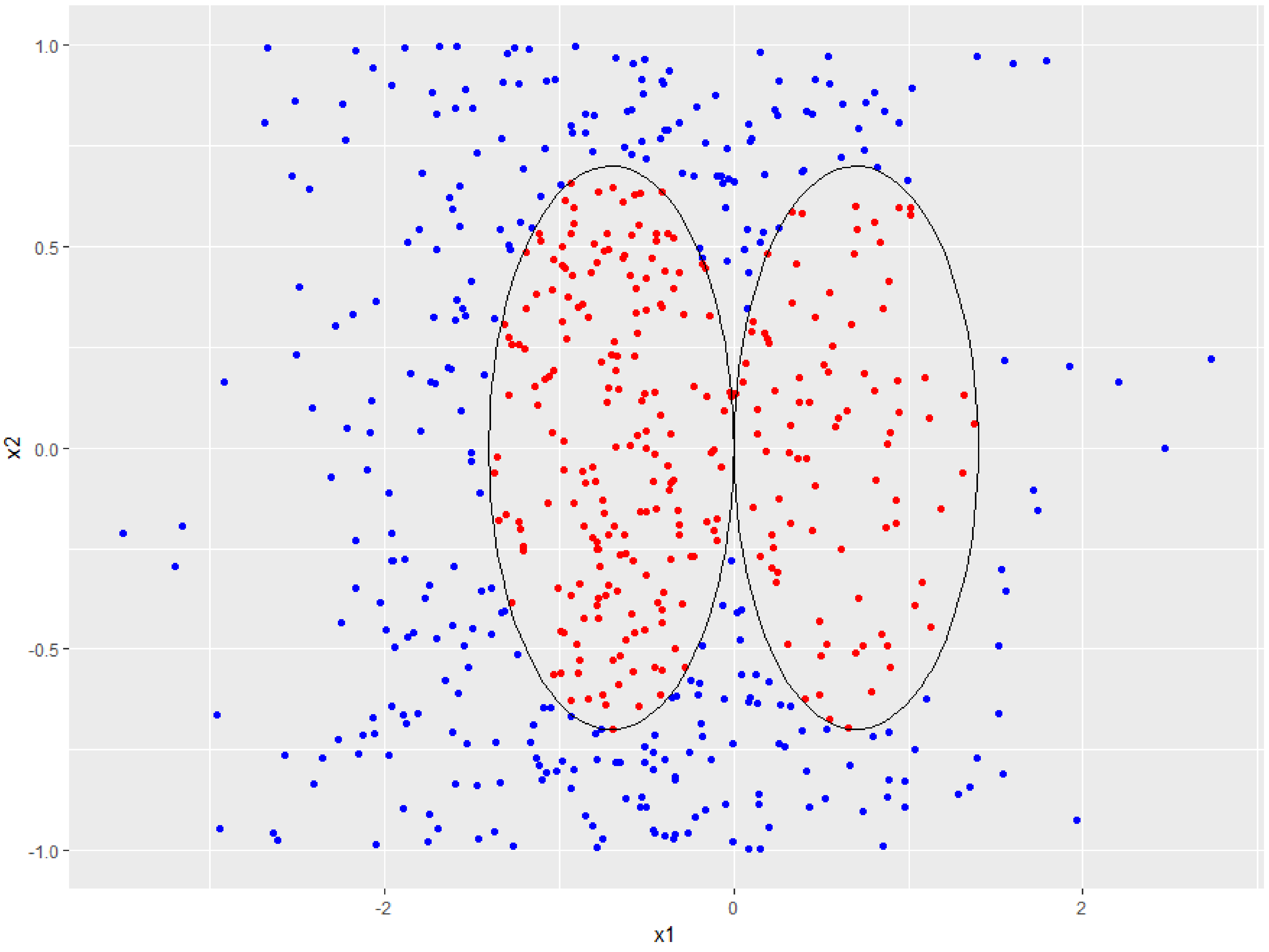
- Visualize the dataset using ggplot; distinguish classes by color

```
library(ggplot2)

p <- ggplot(data = df, aes(x = x1, y = x2, color = y)) +
     geom_point() +
     guides(color = "none") +
     scale_color_manual(values = c("red", "blue"))

p
```



1.0
0.5
0.0
-0.5
-1.0
x2
-2
0
2
x1

```r
# Function to generate points on a circle
circle <- function(x1_center, x2_center, r, npoint = 100) {
   theta <- seq(0, 2 * pi, length.out = npoint)
   x1_circ <- x1_center + r * cos(theta)
   x2_circ <- x2_center + r * sin(theta)
   data.frame(x1c = x1_circ, x2c = x2_circ)
}
# Generate boundary and plot it
boundary_1 <- circle(x1_center = center_1[1], x2_center = center_1[2], r = radius)
p <- p +
     geom_path(data = boundary_1,
               aes(x = x1c, y = x2c),
               inherit.aes = FALSE)
boundary_2 <- circle(x1_center = center_2[1], x2_center = center_2[2], r = radius)
p <- p +
     geom_path(data = boundary_2,
               aes(x = x1c, y = x2c),
               inherit.aes = FALSE)
p
```


1.0
0.5
0.0
-0.5
-1.0
x2
-2
0
2
x1

Time to practice!

SUPPORT VECTOR MACHINES IN R

Motivating the RBF kernel

SUPPORT VECTOR MACHINES IN R

Kailash Awati
Instructor

Quadratic kernel (default parameters)

- Partition data into test/train (not shown)
- Use degree 2 polynomial kernel (default params)

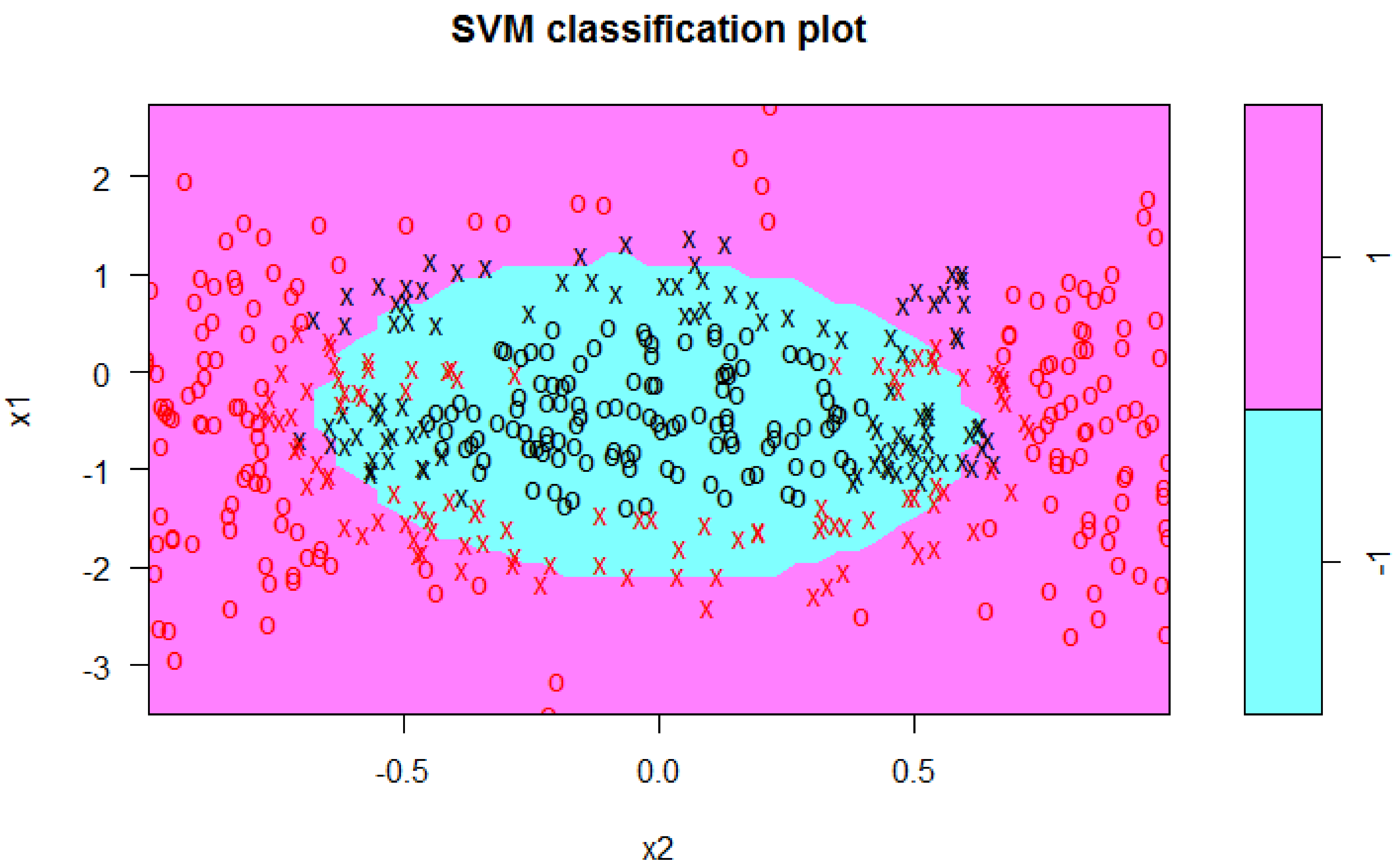
```
svm_model <- svm(y ~ ., data = trainset,
                 type = "C-classification",
                 kernel = "polynomial",
                 degree = 2)
svm_model
```

```
....
Number of Support Vectors:  204
```

```
# Predictions
pred_test <- predict(svm_model, testset)
mean(pred_test == testset$y)
```

```
0.8666667
```

```
plot(svm_model, trainset)
```


SVM classification plot
2
1
0
-1
-2
-3
x1
-0.5
0.0
0.5
x2
1
-1

Try higher degree polynomial

- Rule out odd degrees -3,5,9 etc.
- Try degree 4

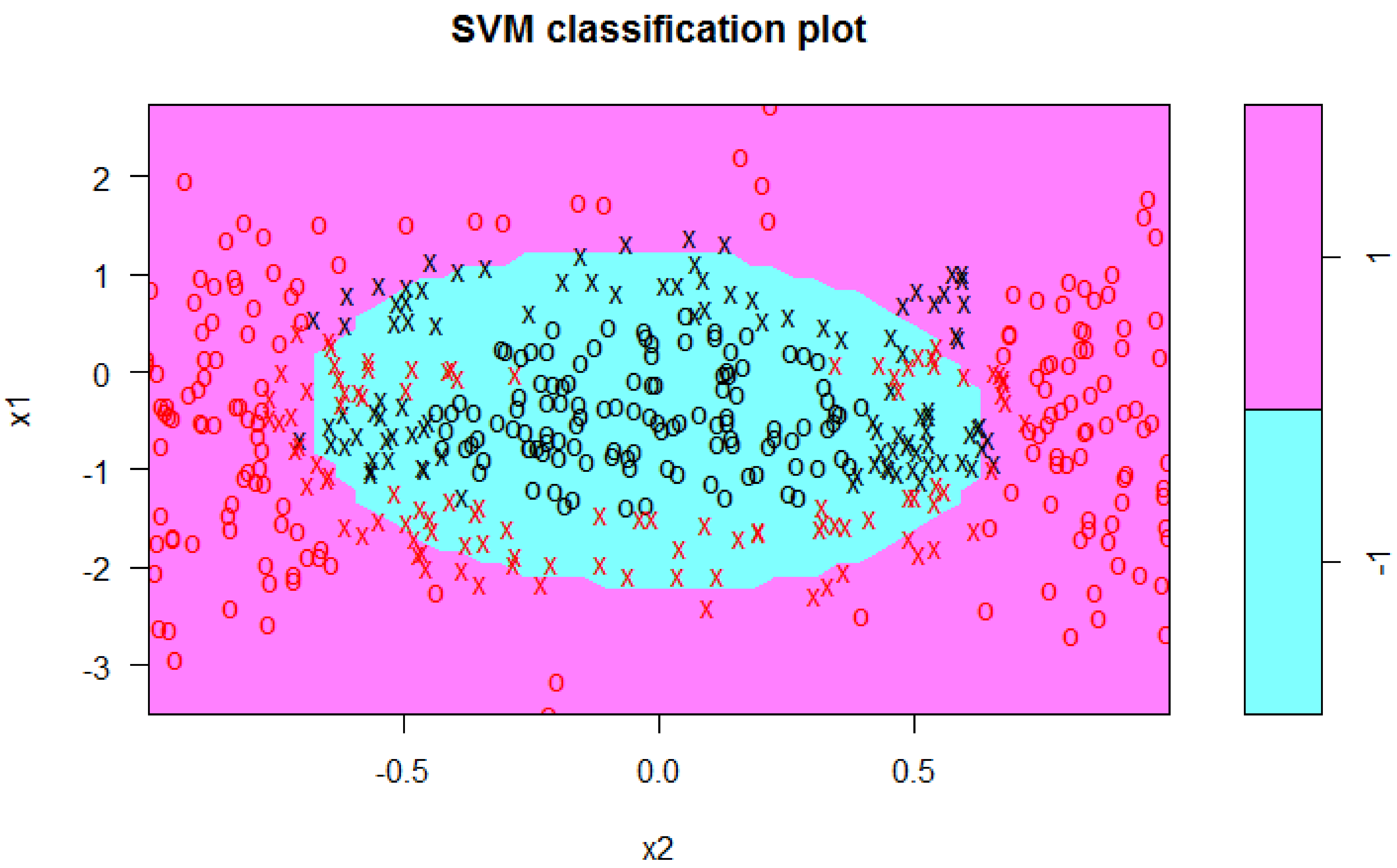
```
svm_model <- svm(y ~ ., data = trainset,
                 type = "C-classification",
                 kernel = "polynomial",
                 degree = 4)
svm_model
```

```
...
Number of Support Vectors: 203
```

```
# Predictions
pred_test <- predict(svm_model, testset)
mean(pred_test == testset$y)
```

```
0.8583333
```

```
plot(svm_model, trainset)
```

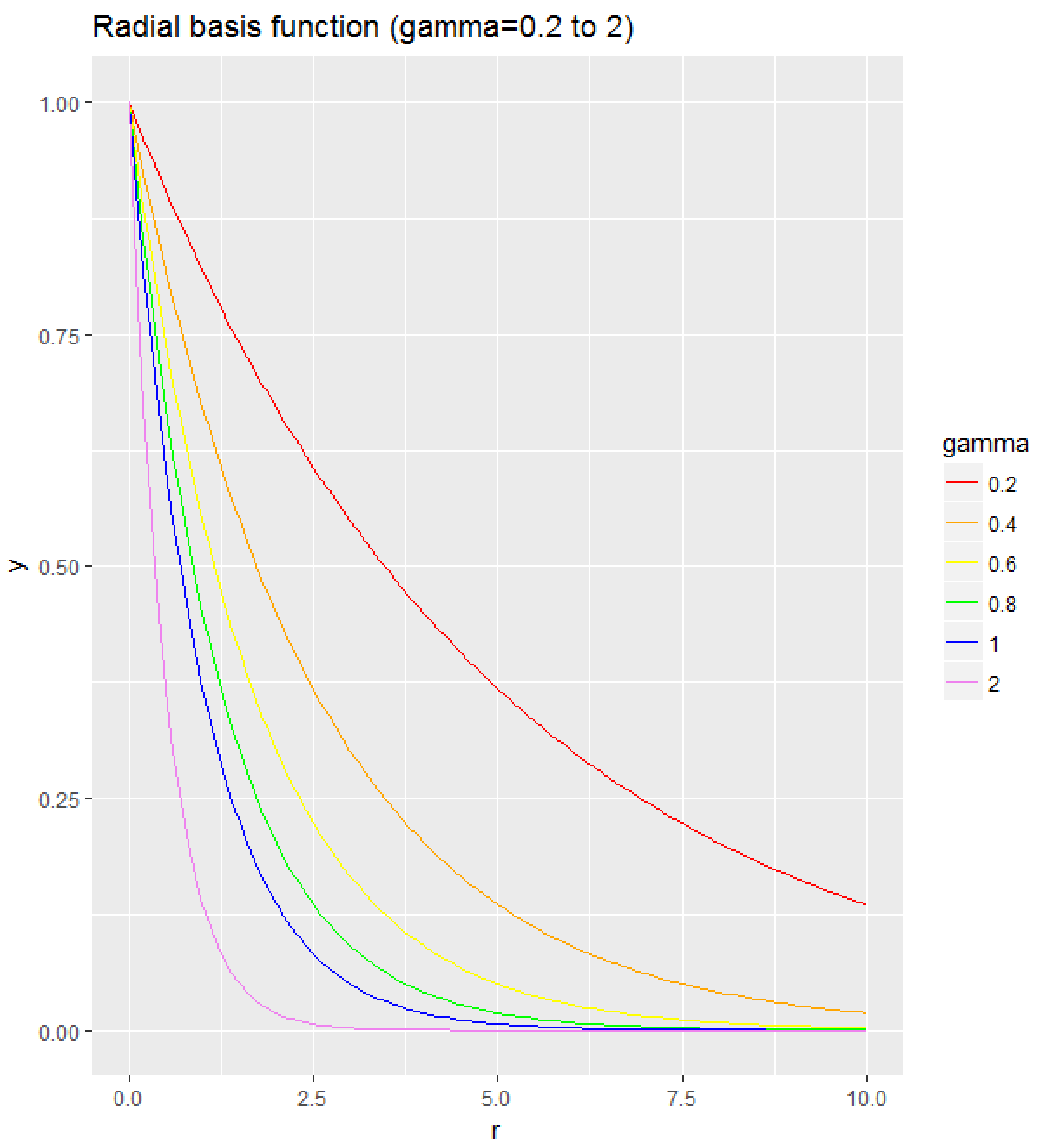

SVM classification plot
x1
x2
2
1
0
-1
-2
-3
-0.5
0.0
0.5
1
-1

Another approach

- **Heuristic:** points close to each other have the same classification:
  - Akin to K-Nearest Neighbors algorithm.
- For a given point in the dataset, say **X1** = (a, b):
  - The kernel should have a maximum at (a, b)
  - Should decay as one moves away from (a, b)
  - The rate of decay should be the same in all directions
  - The rate of decay should be tunable
- A simple function with this property is `exp(-gamma * r)`, where `r` is the distance between X1 and any other point X

How does the RBF kernel vary with gamma (code)

```
#rbf function
rbf <- function(r, gamma) exp(-gamma * r)
ggplot(data.frame(r = c(-0, 10)), aes(r)) +
  stat_function(fun = rbf, args = list(gamma = 0.2), aes(color = "0.2")) +
  stat_function(fun = rbf, args = list(gamma = 0.4), aes(color = "0.4")) +
  stat_function(fun = rbf, args = list(gamma = 0.6), aes(color = "0.6")) +
  stat_function(fun = rbf, args = list(gamma = 0.8), aes(color = "0.8")) +
  stat_function(fun = rbf, args = list(gamma = 1), aes(color = "1")) +
  stat_function(fun = rbf, args = list(gamma = 2), aes(color = "2")) +
  scale_color_manual("gamma",
                     values = c("red","orange","yellow",
                                "green","blue","violet")) +
  ggtitle("Radial basis function (gamma = 0.2 to 2)")
```


Radial basis function (gamma=0.2 to 2)
1.00
0.75
0.50
0.25
0.00
y
0.0
2.5
5.0
7.5
10.0
r
gamma
0.2
0.4
0.6
0.8
1
2

Time to practice!

SUPPORT VECTOR MACHINES IN R

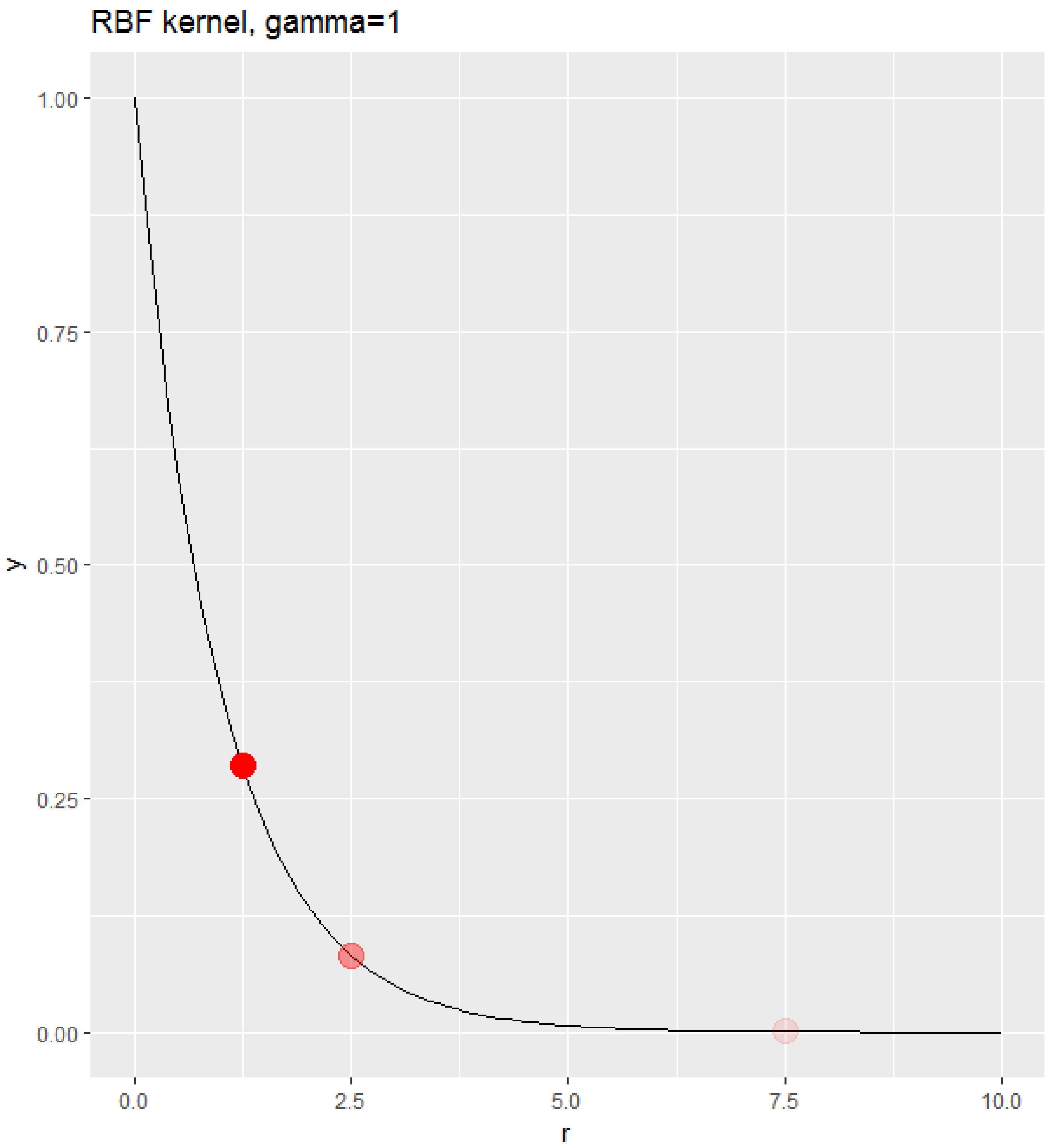
The RBF Kernel

SUPPORT VECTOR MACHINES IN R

Kailash Awati
Instructor

RBF Kernel in a nutshell

- Decreasing function of distance between two points in dataset.
- Simulates k-NN algorithm.



RBF kernel, gamma=1
1.00
0.75
0.50
0.25
0.00
y
0.0
2.5
5.0
7.5
10.0
r

Building an SVM using the RBF kernel

- Build RBF kernel SVM for complex dataset

```
svm_model <- svm(y ~ .,
                data = trainset,
                type = "C-classification",
                kernel = "radial")
```

- Calculate training/test accuracy and plot against training dataset.

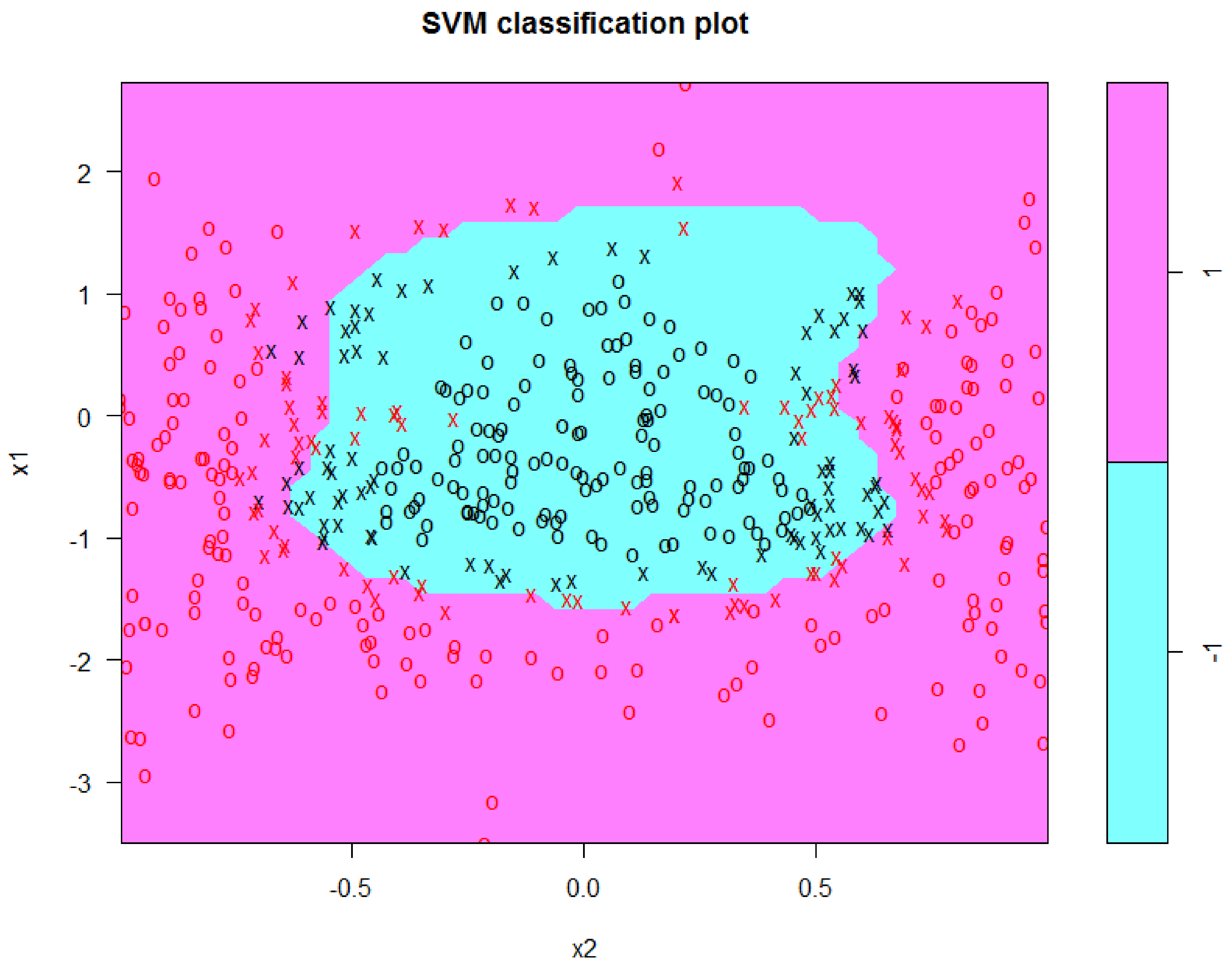
```
pred_train <- predict(svm_model, trainset)
mean(pred_train == trainset$y)
```

```
0.93125
```

```
pred_test <- predict(svm_model, testset)
mean(pred_test == testset$y)
```

```
0.9416667
```

```
#plot decision boundary
plot(svm_model, trainset)
```



SVM classification plot
x1
x2
2
1
0
-1
-2
-3
-0.5
0.0
0.5
1
-1

Refining the decision boundary

- Tune `gamma` and `cost` using `tune.svm()`

```
# Tune parameters
tune_out <- tune.svm(x = trainset[,-3],
                     y = trainset[,3],
                     gamma = 5 * 10 ^ (-2:2),
                     cost = c(0.01, 0.1, 1, 10, 100
                     type = "C-classification",
                     kernel = "radial")
```

- Print best parameters

```
# Print best values of cost and gamma
tune_out$best.parameters$cost
```

```
1
```

```
tune_out$best.parameters$gamma
```

```
5
```

The tuned model

- Build tuned model using `best.parameters`

```
svm_model <- svm(y ~ ., data = trainset,
              type = "C-classification",
              kernel = "radial",
              cost = tune_out$best.parameters$cost,
              gamma = tune_out$best.parameters$gamma)
```

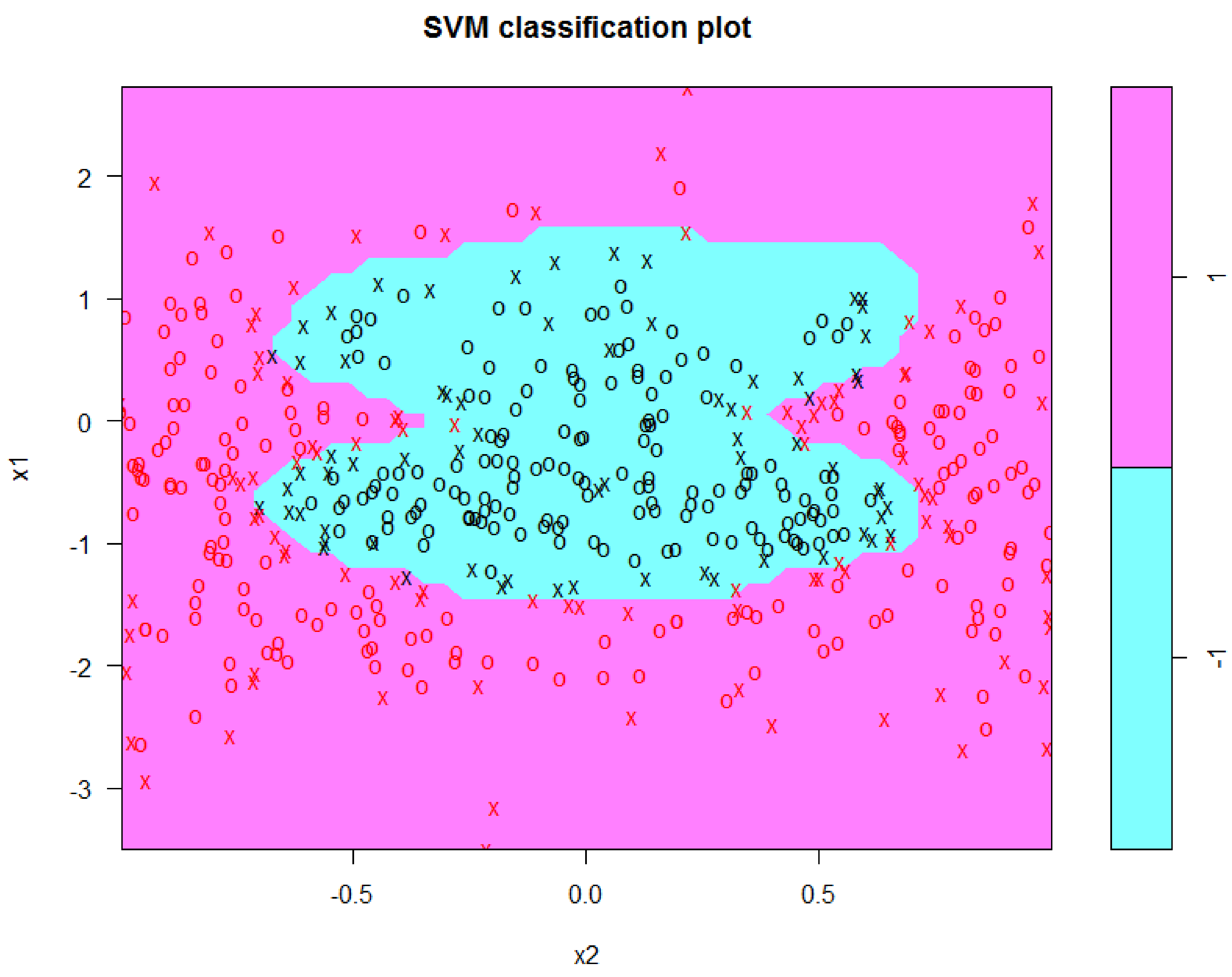
- Calculate test accuracy

```
mean(pred_test == testset$y)
```

```
0.95
```

- Plot decision boundary

```
plot(svm_model, trainset)
```



SVM classification plot
x1
x2
2
1
0
-1
-2
-3
-0.5
0.0
0.5
1
-1

Time to practice!

SUPPORT VECTOR MACHINES IN R