



SUPPORT VECTOR MACHINES IN R

Linear Support Vector Machines

Split into training and test sets

- The dataset generated in previous chapter is in dataframe df.
- Split dataset into training and test sets
- Random 80/20 split

```
#set seed for reproducibility
set.seed() = 1
#assign rows to training/test sets randomly in 80/20 proportion
df[, "train"] <- ifelse(runif(nrow(df)) < 0.8, 1, 0)
```

```
#separate training and test sets
trainset <- df[df$train==1,]
testset <- df[df$train==0,]
trainColNum <- grep("train", names(trainset))
trainset <- trainset[, -trainColNum]
testset <- testset[, -trainColNum]
```



Decision boundaries and kernels

- Decision boundaries can have different shapes - lines, polynomials or more complex functions.
- Type of decision boundary is called a **kernel**.
- Kernel must be specified upfront.
- This chapter focuses on linear kernels.

SVM with Linear Kernel

- We'll use the `svm` function from the `e1071` library.
- The function has a number of parameters. We'll set the following explicitly:
 - **formula** - a formula specifying the dependent variable. `y` in our case.
 - **data** - dataframe containing the data - i.e. trainset.
 - **type** - set to `C-classification(classification problem)`.
 - **kernel** - this is the form of the decision boundary, linear in this case.
 - **cost** and **gamma** - these are parameters that are used to tune the model

Building a Linear SVM

- Load e1071 library and invoke svm() function

```
library(e1071)
```

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

Overview of model

- Entering `svm_model` gives:
 - an overview of the model including classification and kernel type
 - tuning parameter values

```
svm_model
```

Call:

```
svm(formula = y ~ .,  
     data = trainset,  
     type = "C-classification",  
     kernel = "linear",  
     scale = FALSE)
```

Parameters:

```
SVM-Type: C-classification  
SVM-Kernel: linear  
cost: 1  
gamma: 0.5
```

Number of Support Vectors: 55



Exploring the Model

```
#index of support vectors in training dataset
svm_model$index
[1] 4 8 10 11 18 37 38 39 47 59 60 74 76 77 78 80 83 ...
      -- -- --

#support vectors
svm_model$SV
      x1      x2
5 0.519095949 0.44232464
      -- -- --

#negative intercept (unweighted)
svm_model$rho
[1] -0.1087075

#weighting coefficients for support vectors
svm_model$coefs
      [,1]
[1,] 1.0000000
      -- -- --
```

Model Accuracy

- Obtain class predictions for training and test sets.
- Evaluate the training and test set accuracy of the model.

```
#training accuracy  
pred_train <- predict(svm_model,trainset)  
mean(pred_train==trainset$y)  
[1] 1
```

```
#test accuracy  
pred_test <- predict(svm_model,testset)  
mean(pred_test==testset$y)  
[1] 1  
#perfect!!
```




SUPPORT VECTOR MACHINES IN R

Time to practice!



SUPPORT VECTOR MACHINES IN R

Visualizing linear SVMs

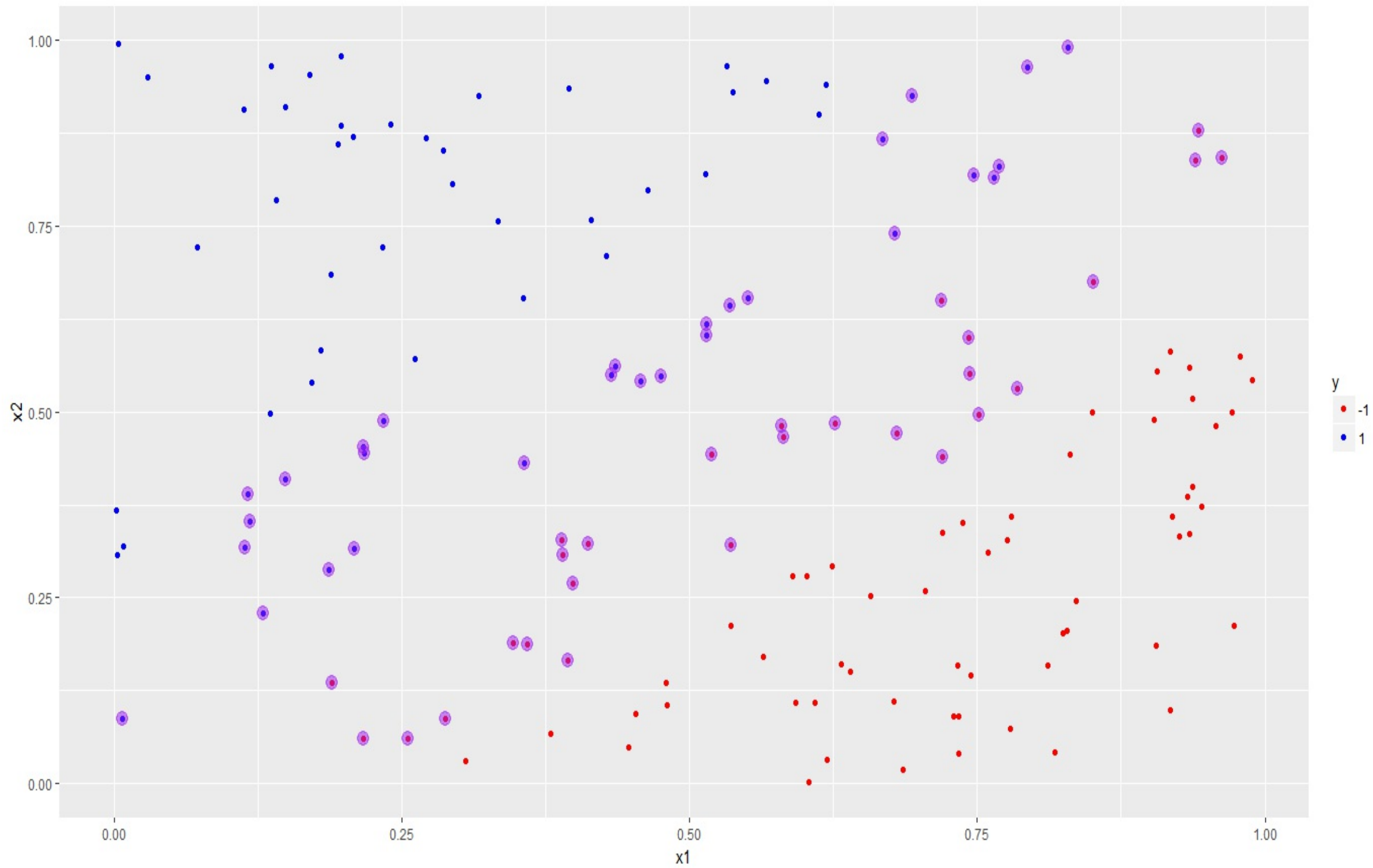
Visualizing support vectors

- Plot the training data using `ggplot()`.

```
#visualize training data, distinguish classes using color
p <- ggplot(data = trainset, aes(x = x1, y = x2, color = y)) +
  geom_point() +
  scale_color_manual(values = c("red", "blue"))
#render plot
p
```

- Mark out the support vectors using index from `svm_model`.

```
#identify support vectors
df_sv <- trainset[svm_model$index,]
#mark out support vectors in plot
p <- p + geom_point(data = df_sv,
  aes(x = x1, y = x2),
  color = "purple",
  size = 4, alpha = 0.5)
#render plot
p
```



Slope and intercept of the decision boundary

Find slope and intercept of the boundary:

- Build the weight vector, w , from coefs and SV elements of `svm_model`.

```
#build weight vector  
w <- t(svm_model$coefs) %*% svm_model$SV
```

- $\text{slope} = -w[1]/w[2]$

```
#calculate slope and save it to a variable  
slope_1 <- -w[1]/w[2]
```

- $\text{intercept} = \text{svm_model}\$rho/w[2]$

```
#calculate intercept and save it to a variable  
intercept_1 <- svm_model$rho/w[2]
```

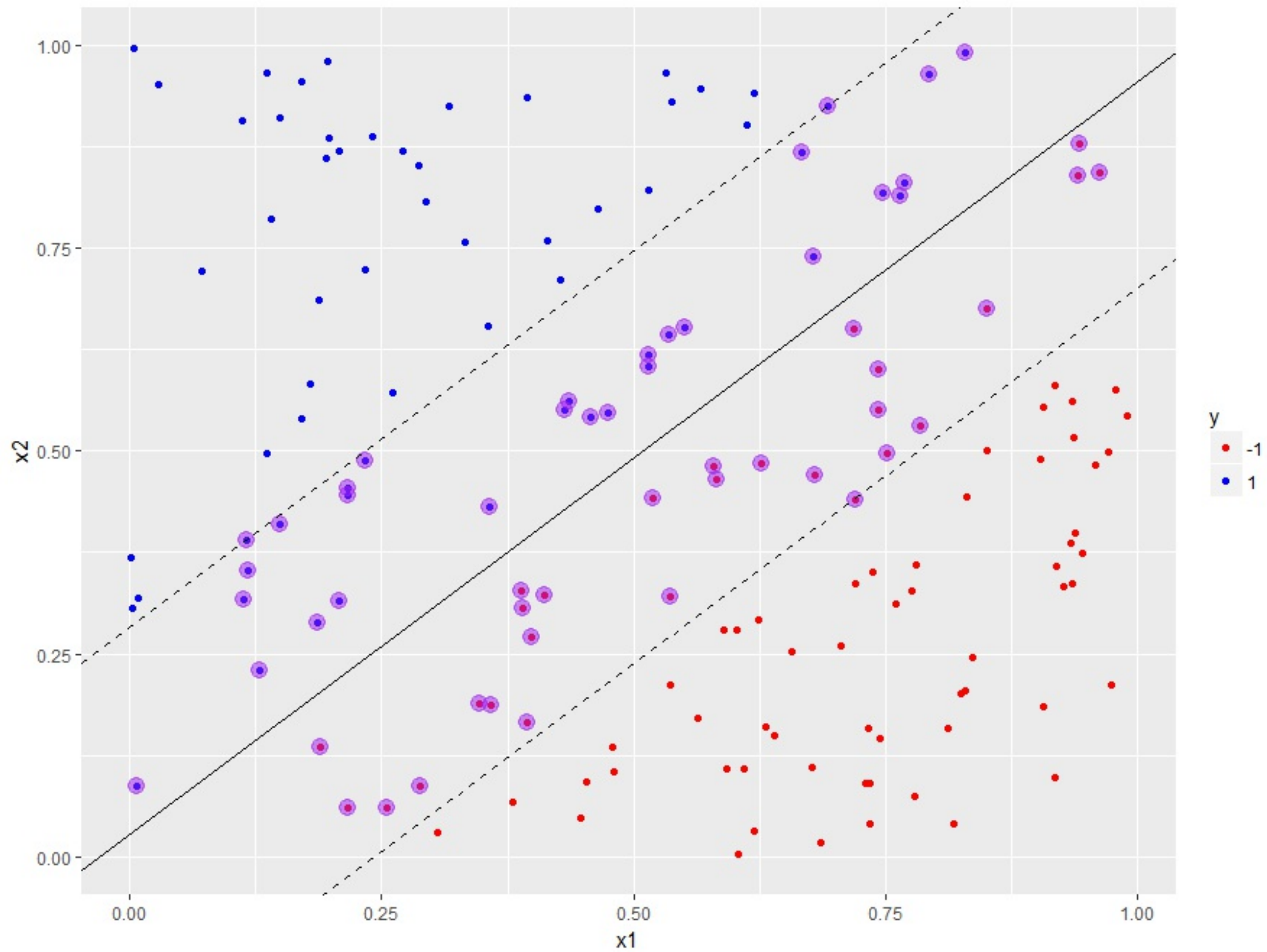
Visualizing the decision and margin boundaries

- Add decision boundary using slope and intercept calculated in previous slide.
- We use `geom_abline()` to add the decision boundary to the plot.

```
#plot decision boundary based on calculated slope and intercept
p <- p + geom_abline(slope = slope_1,
                    intercept = intercept_1)
```

- Margins parallel to decision boundary, offset by $1/w[2]$ on either side of it.

```
#add margins to plot
p <- p +
  geom_abline(slope = slope_1,
              intercept = intercept_1 - 1/w[2],
              linetype = "dashed") +
  geom_abline(slope = slope_1,
              intercept = intercept_1 + 1/w[2],
              linetype = "dashed")
#display plot
```





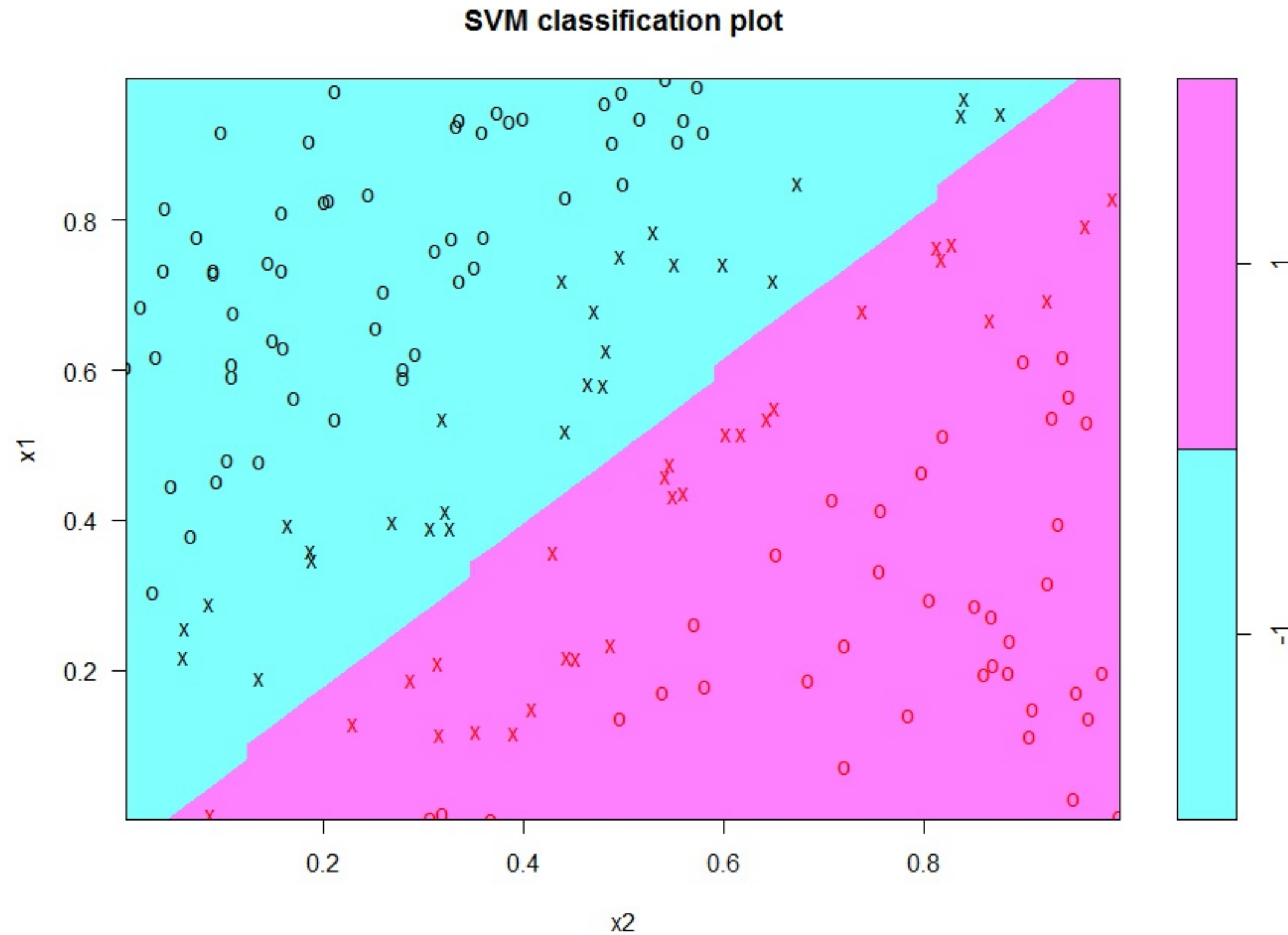
Soft margin classifiers

- Allow for uncertainty in location / shape of boundary
 - Never perfectly linear
 - Usually unknown
- Our decision boundary is linear, so we can reduce margin

Visualizing the decision boundary using the `svm plot()` function

- The `svm plot()` function in `e1071` offers an easy way to plot the decision boundary.

```
#visualize decision boundary using built in plot function
plot(x = svm_model,
     data = trainset)
```





SUPPORT VECTOR MACHINES IN R

Time to practice!



SUPPORT VECTOR MACHINES IN R

Tuning linear SVMs

Linear SVM, default cost

```
library(e1071)
svm_model<- svm(y ~ .,
                data = trainset,
                type = "C-classification",
                kernel = "linear",
                scale = FALSE)
```

```
#print model summary
```

```
svm_model
```

```
Call:
```

```
svm(formula = y ~ .,
     data = trainset,
     type = "C-classification",
     kernel = "linear",
     scale = FALSE)
```

```
Parameters:
```

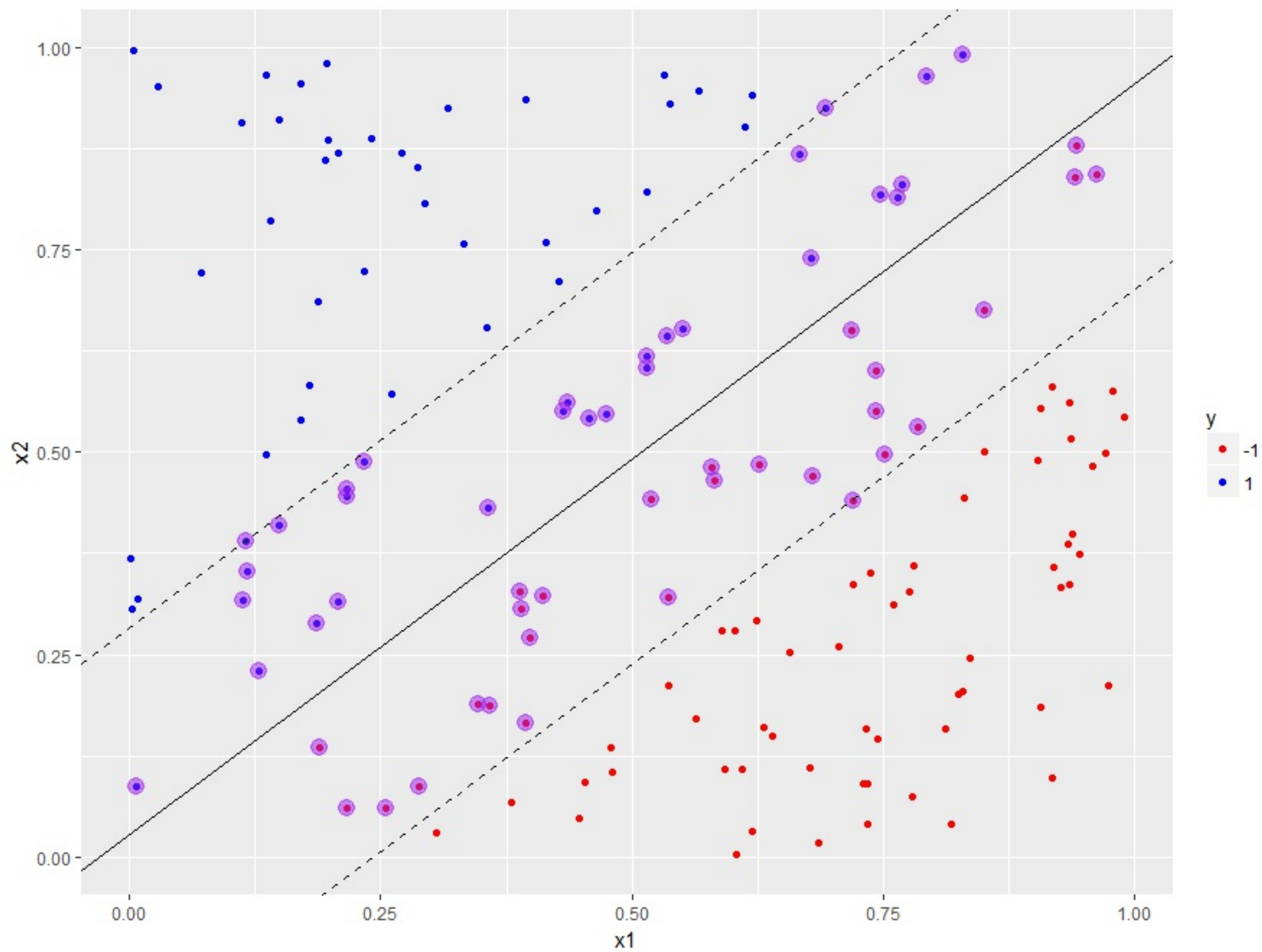
```
SVM-Type: C-classification
```

```
SVM-Kernel: linear
```

```
cost: 1
```

```
gamma: 0.5
```

```
Number of Support Vectors: 55
```



Linear SVM with cost = 100

```
library(e1071)
svm_model<- svm(y ~ .,
                data = trainset,
                type = "C-classification",
                kernel = "linear",
                cost = 100,
                scale = FALSE)
```

```
#print model summary
```

```
svm_model
```

```
Call:
```

```
svm(formula = y ~ .,
     data = trainset,
     type = "C-classification",
     kernel = "linear",
     cost = 100,
     scale = FALSE)
```

```
Parameters:
```

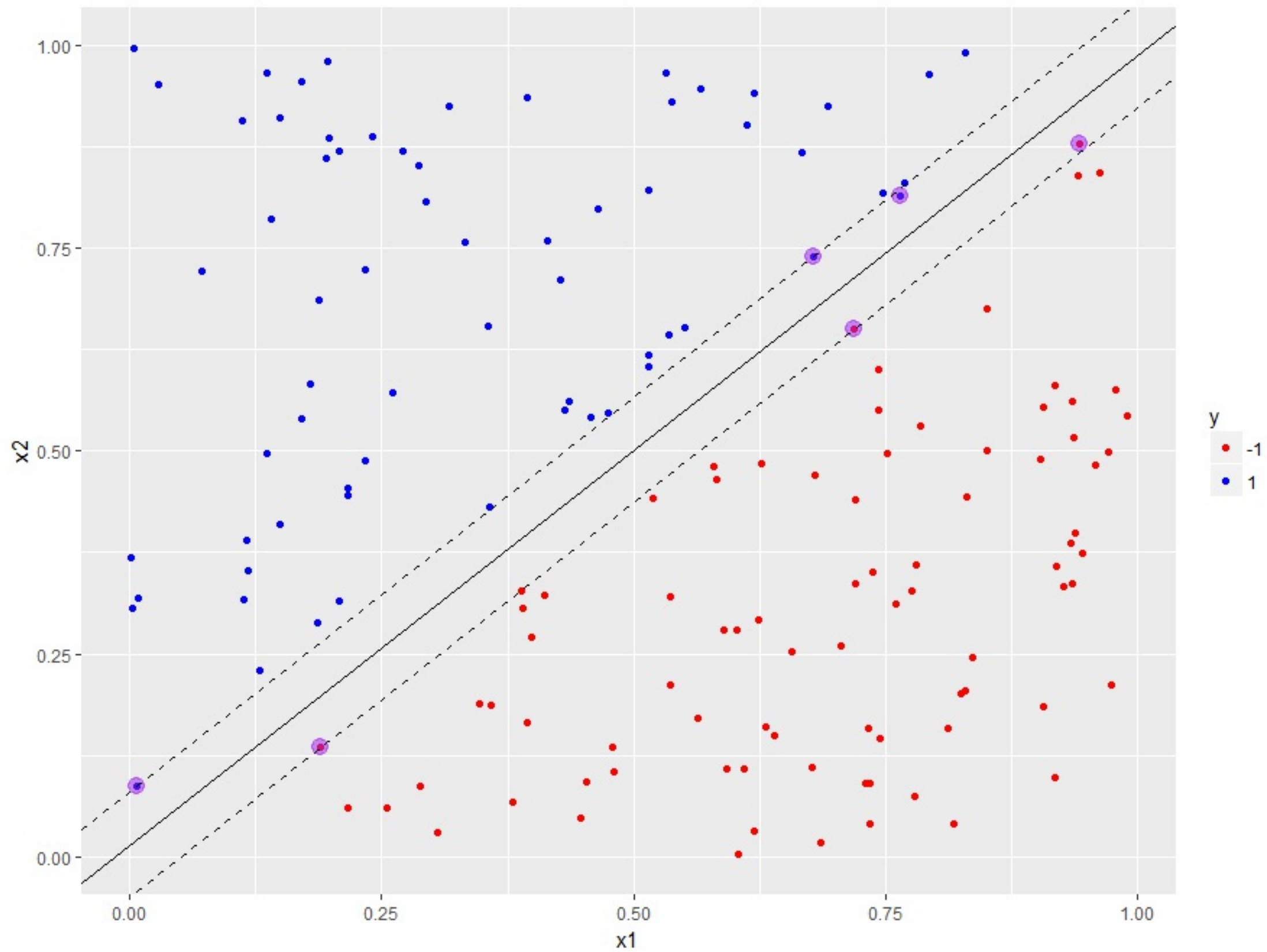
```
SVM-Type: C-classification
```

```
SVM-Kernel: linear
```

```
cost: 100
```

```
gamma: 0.5
```

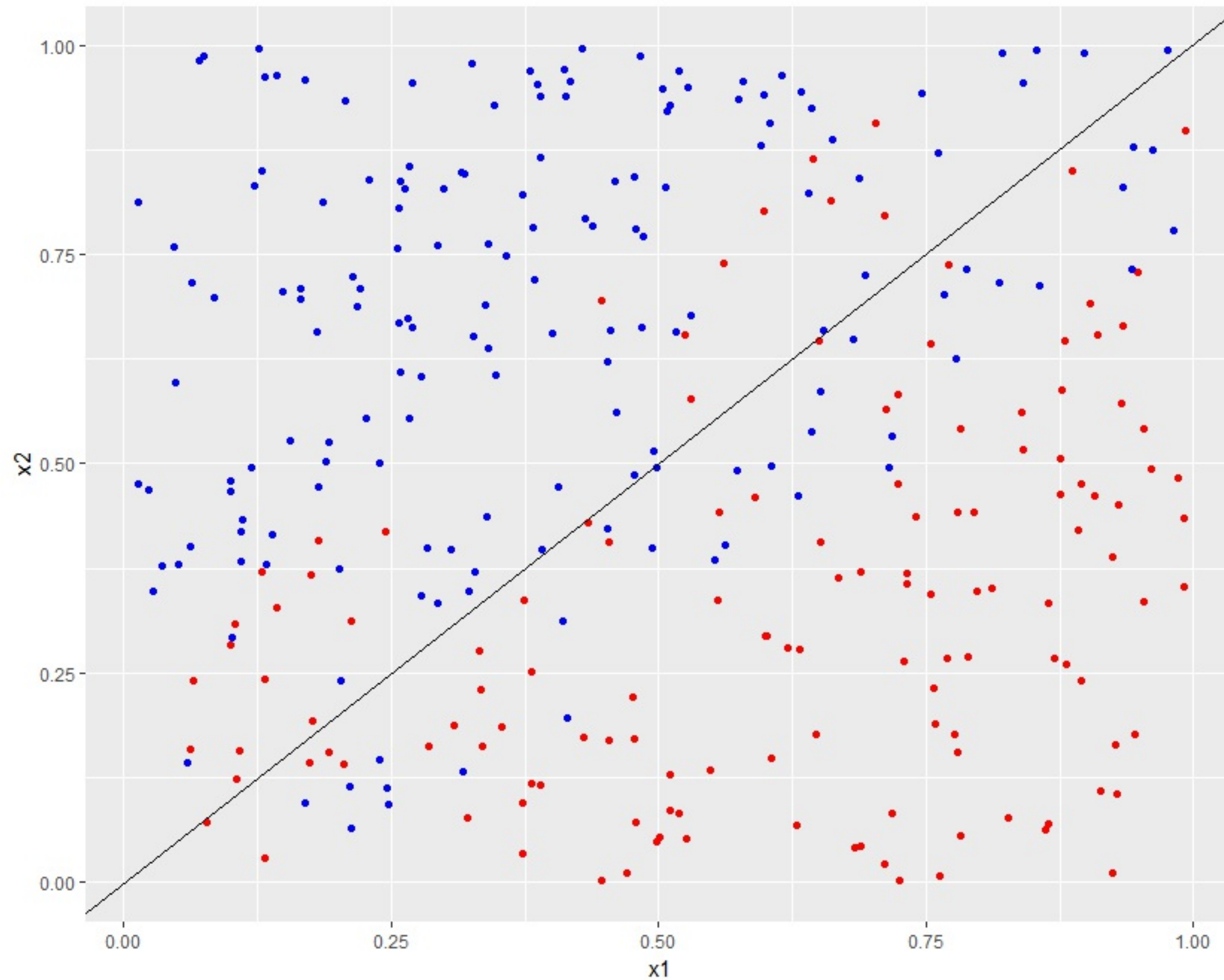
```
Number of Support Vectors: 6
```





Implication

- Can be useful to reduce margin if decision boundary is known to be linear
- ...but this is rarely the case in real life



Nonlinear dataset, linear SVM (cost = 100)

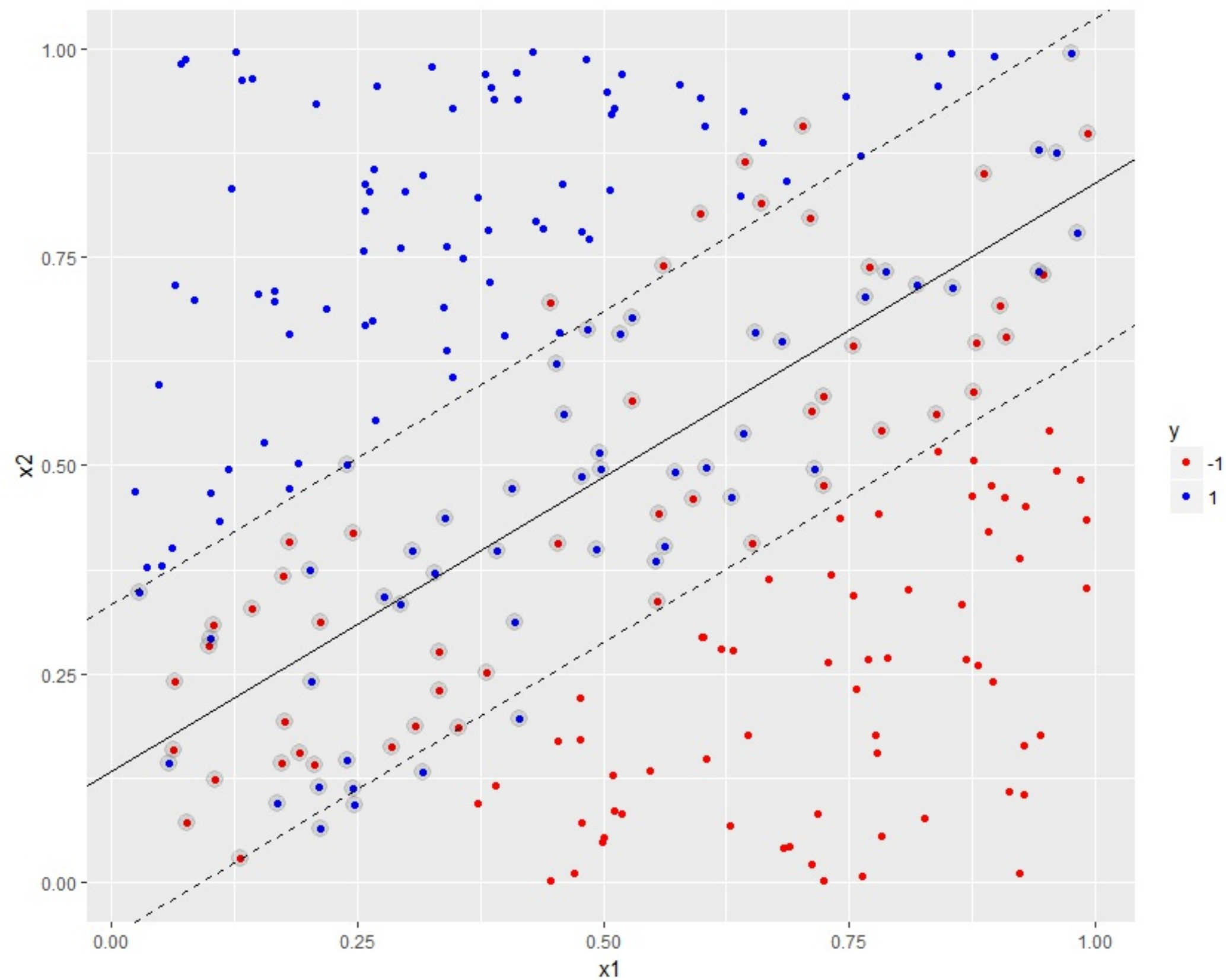
- Build cost=100 model using training set composed of 80% of data

```
#build model
library(e1071)
svm_model<- svm(y ~ .,
                data = trainset,
                type = "C-classification",
                kernel = "linear",
                cost = 100,
                scale = FALSE)
```

- Calculate accuracy

```
#train and test accuracy
pred_train <- predict(svm_model,trainset)
mean(pred_train==trainset$y)
[1] 0.8208333
pred_test <- predict(svm_model,testset)
mean(pred_test==testset$y)
[1] 0.85
```

- Average test accuracy over 50 random train/test splits: 82.9%



Nonlinear dataset, linear SVM (cost = 1)

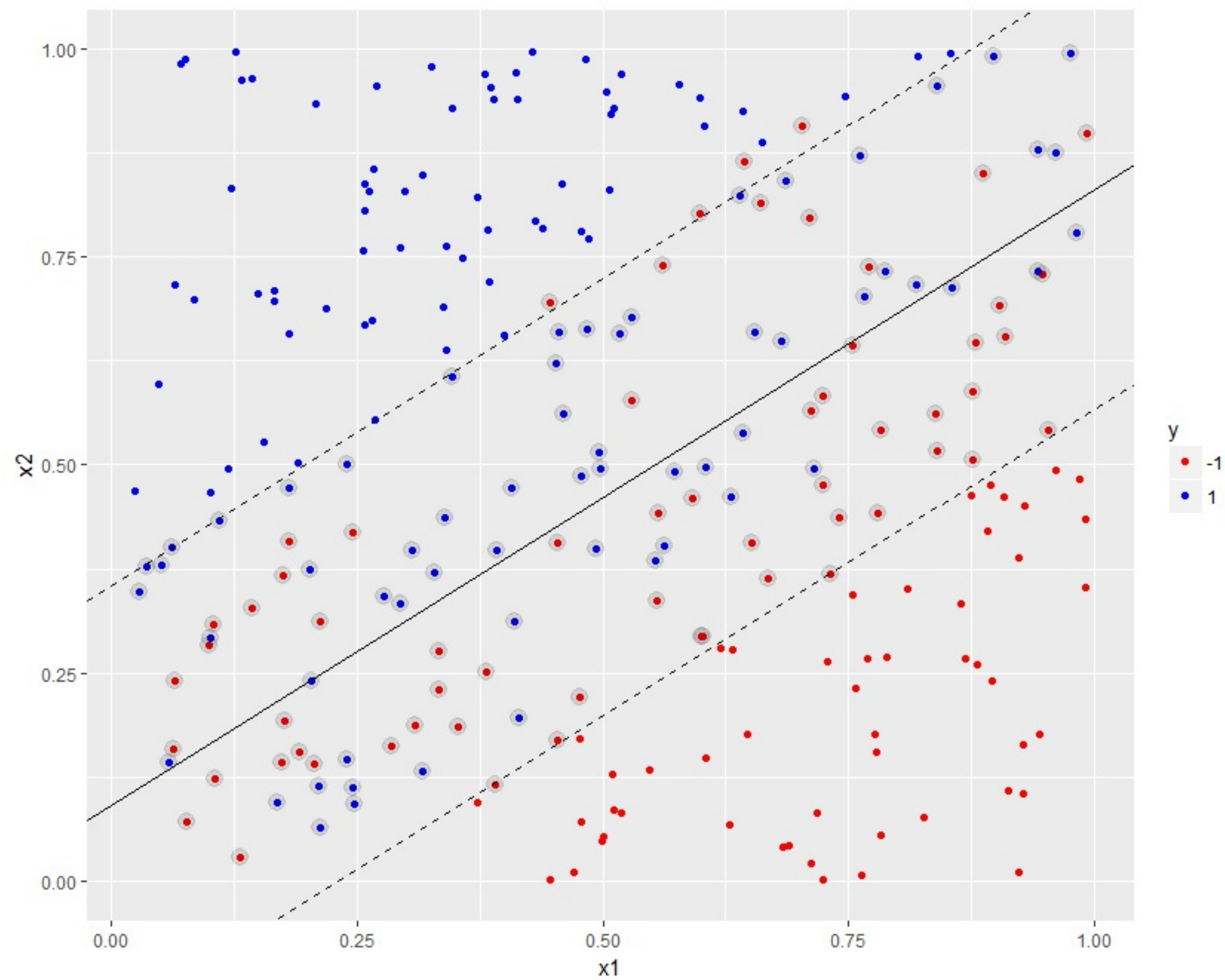
- Rebuild model setting cost = 1

```
#trainset contains 80% of data, same train/test split as before.  
#build model  
svm_model<- svm(y ~ .,  
                data = trainset,  
                type = "C-classification",  
                kernel = "linear",  
                cost = 1,  
                scale = FALSE)
```

- Calculate test accuracy

```
#test accuracy  
pred_test <- predict(svm_model,testset)  
mean(pred_test==testset$y)  
[1] 0.8666667
```

- Average test accuracy over 50 random train/test splits: 83.7%





SUPPORT VECTOR MACHINES IN R

Time to practice!



SUPPORT VECTOR MACHINES IN R

Multiclass problems



The iris dataset - an introduction

- 150 measurements of 5 attributes
 - Petal width and length - number (predictor variables)
 - Sepal width and length - number (predictor variables)
 - Species - category: setosa, virginica or versicolor (predicted variable)
- Dataset available from [UCI ML repository](#)

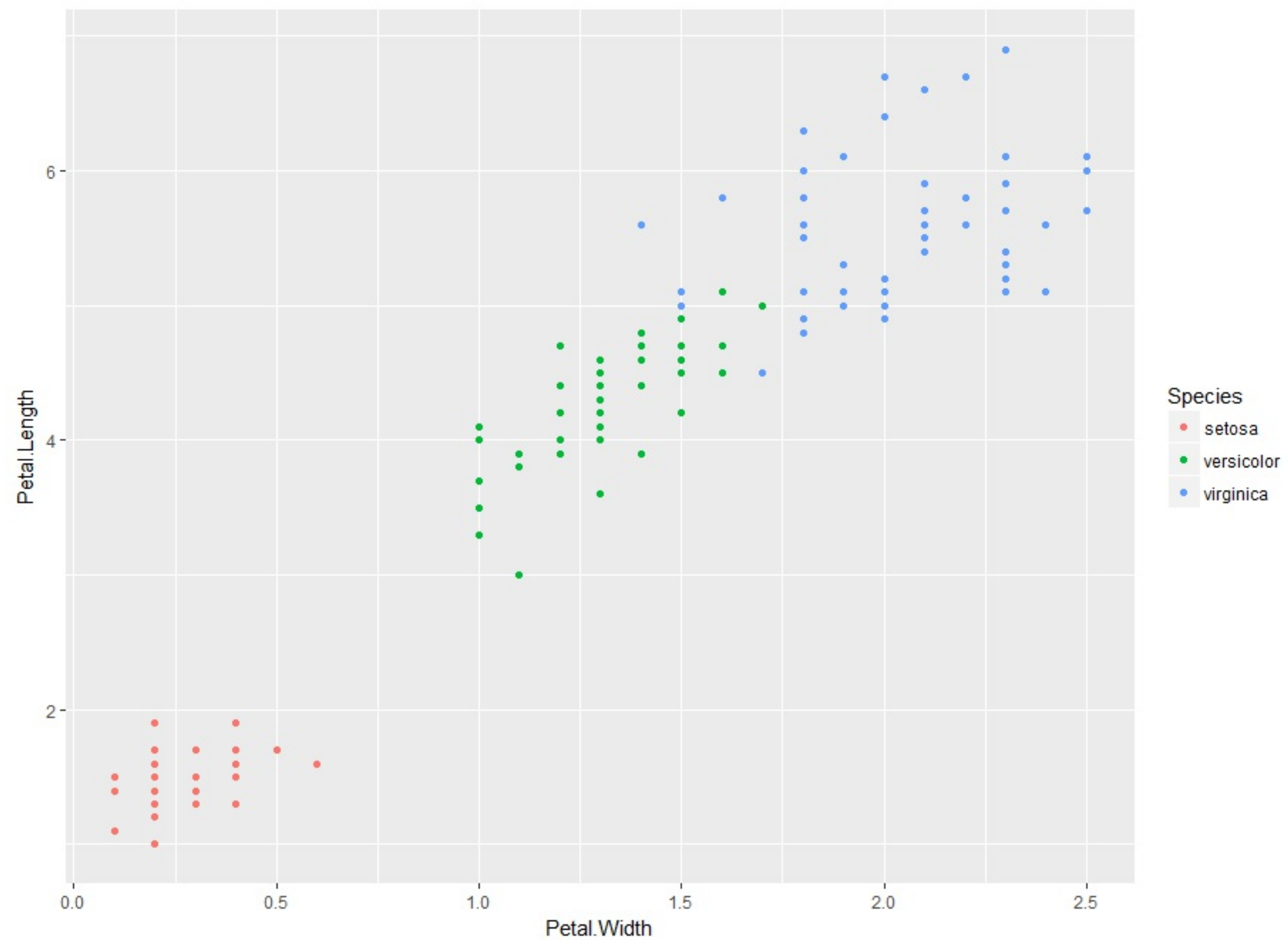
Visualizing the iris dataset

- Plot petal length vs petal width.

```
library(ggplot2)

#plot petal length vs width for dataset, distinguish species by color
p <- ggplot(data = iris,
            aes(x = Petal.Width,
                y = Petal.Length,
                color = Species)) +
  geom_point()

#display plot
p
```





How does the SVM algorithm deal with multiclass problems?

- SVMs are essentially binary classifiers.
- Can be applied to multiclass problems using the following voting strategy:
 - Partition the data into subsets containing two classes each.
 - Solve the binary classification problem for each subset.
 - Use majority vote to assign a class to each data point.
- Called **one-against-one** classification strategy.

Building a multiclass linear SVM

- Build a linear SVM for the iris dataset
 - 80/20 training / test split (seed 10), default cost

```
library(e1071)

#build model
svm_model<- svm(Species ~ .,
                data = trainset,
                type = "C-classification",
                kernel = "linear")
```

- Calculate accuracy

```
#accuracy
pred_train <- predict(svm_model,trainset)
mean(pred_train==trainset$Species)
[1] 0.9756098
pred_test <- predict(svm_model,testset)
mean(pred_test==testset$Species)
[1] 0.962963
```



SUPPORT VECTOR MACHINES IN R

Time to practice!