NBA 4920/6921 Lecture 21

Support Vector Machines

Murat Unal

11/11/2021

```r
rm(list=ls())
options(digits = 3, scipen = 999)
library(tidyverse)
library(ggplot2)
library(ISLR)
library(lmtest)
library(sandwich)
library(jtools)
library(caret)
library(ROCR)
library(e1071)
library(GGally)
set.seed(1)
```

Support Vector Machines

- Are a general class of classifiers that essentially attempt to separate two classes of observations
- The support vector machine generalizes a much simpler classifier—the maximal margin classifier
- The maximal margin classifier attempts to separate the two classes in our prediction space using a single hyperplane.

What's a hyperplane?

- A hyperplane is a dimensional subspace that is flat (no curvature)
- In $p = 1$ dimensions, a hyperplane is a point
- In $p = 2$ dimensions, a hyperplane is a line.
- In $p = 3$ dimensions, a hyperplane is a plane.

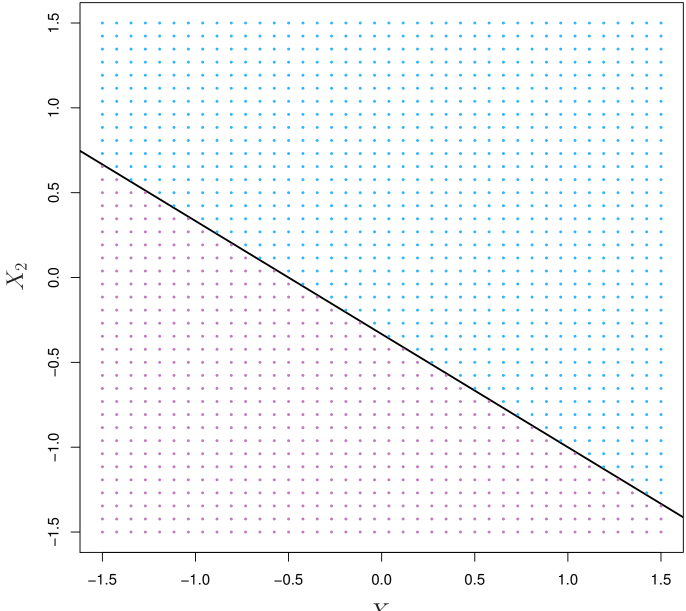
Hyperplanes

- We can define a hyperplane in p dimensions by constraining the linear combination of the p dimensions.
- For example, in two dimensions a hyperplane is defined by $\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$ which is just the equation for a line.
- The points $X = (X_1, X_2)$ that satisfy the equality live on the hyperplane.

Separating hyperplanes

- More generally, in p dimensions, we define a hyperplane by $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0$
- If $X = (X_1, X_2, \ldots, X_p)$ satisfies the equality, it is on the hyperplane
- If $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p > 0$, then X is **above** the hyperplane
- If $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p < 0$, then X is **below** the hyperplane
- The hyperplane separates the p-dimensional space into two **halves**

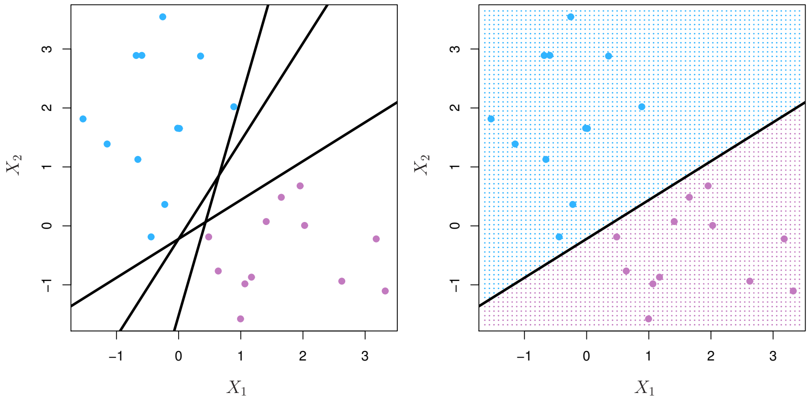
- A separating hyperplane in two dimensions: $1 + 2X_1 + 3X_2 = 0$



Separating hyperplanes and classification

- To make a prediction for observation (x^0, y^0)
- We classify points that live **above** of the plane as one class
- If $f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p > 0$, then $\hat{y}^0 = Class1$
- We classify points that live **below** of the plane as one class
- If $f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p < 0$, then $\hat{y}^0 = Class2$
- This strategy assumes a separating hyperplane exists

- If a separating hyperplane exists, then it defines a binary classifier.
- Moreover, many separating hyperplanes exist.



- We can also make use of the magnitude of $f(x^0)$
- If $f(x^0)$ is far from zero, then this means that x^0 lies far from the hyperplane, and so we can be confident about our class assignment for x^0.
- On the other hand, if $f(x^0)$ is close to zero, then this means that x^0 is located near the hyperplane, and so we are less certain about the class assignment for x^0.

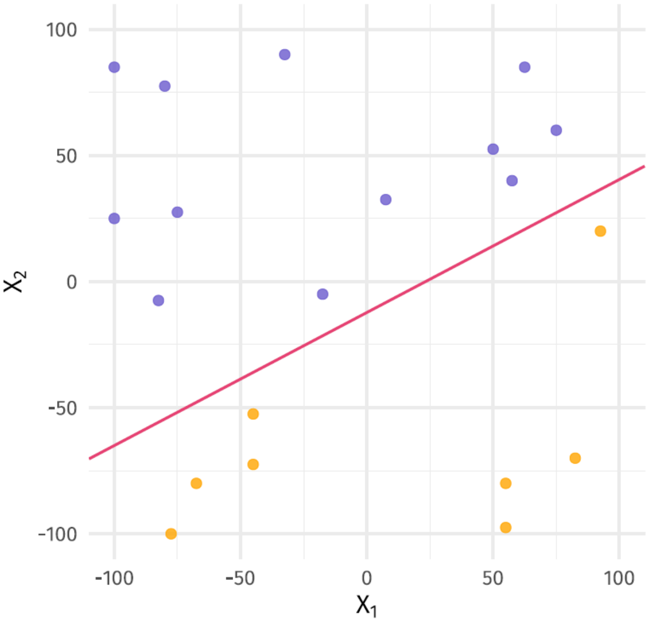
Which separating hyperplane

- How do we choose between the possible hyperplanes?
- One solution: Choose the separating hyperplane that is **farthest** from the training data points—maximizing separation.

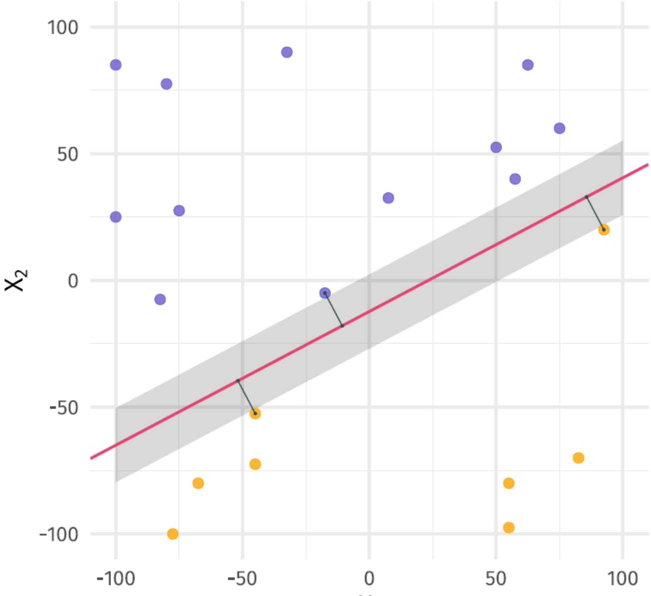
The maximal margin hyperplane

- Separates the two classes of obsevations
- Maximizes the **margin**—the distance to the nearest observation, where distance is a point's perpendicular distance to the hyperplane

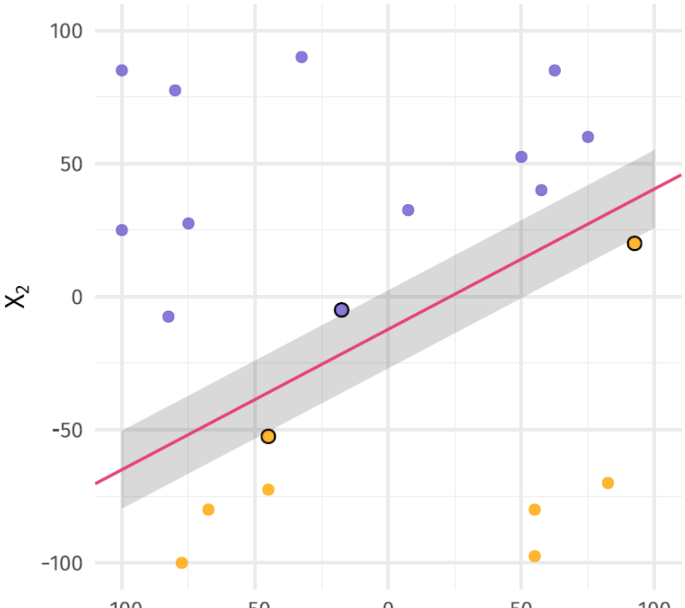
The maximal margin hyperplane...



. . . maximizes the margin between the hyperplane and training data. . .



. . . and is supported by three equidistant observations—the support vectors.



- Formally, the maximal margin hyperplane solves the problem:
- Maximize the margin M over the set of $\beta_0, \beta_1, \cdots, \beta_p, M$ such that

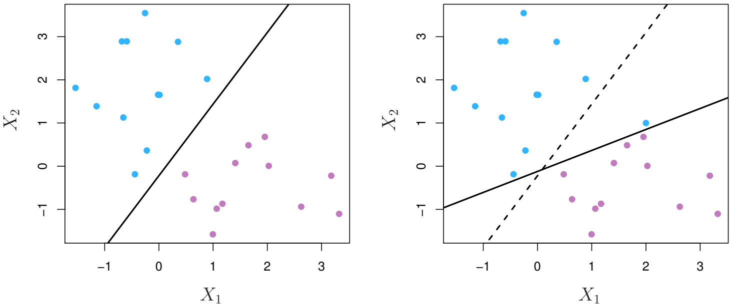
$$\sum_{j=1}^{p} \beta_j^2 = 1 \tag{1}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M \tag{2}$$

- (2) Ensures we separate (classify) observations correctly.
- (1) Allows us to interpret (2) as "distance from the hyperplane"

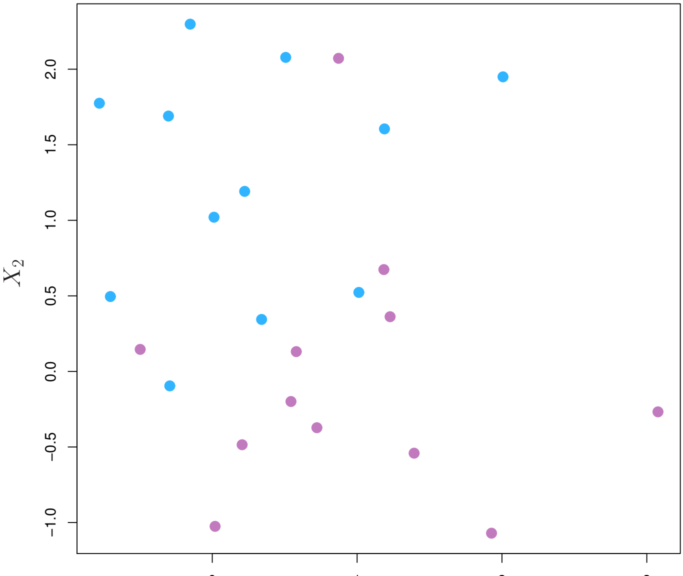
The maximal margin classifier

- The maximal margin hyperplane produces the maximal margin classifier
- The decision boundary only uses the support vectors—very sensitive



- This classifier can struggle in large dimensions

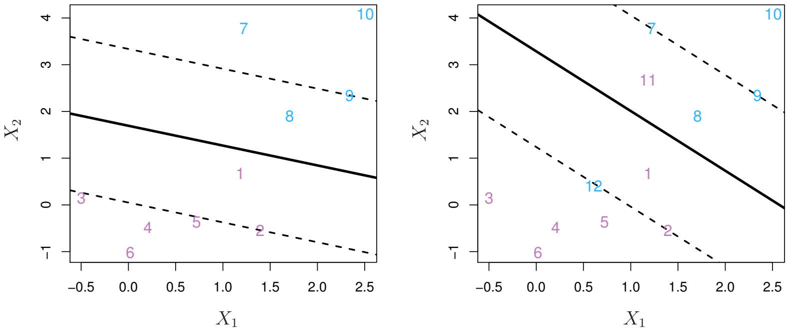
- In many cases no separating hyperplane exists, and so there is no maximal margin classifier



- However, we can extend the concept of a separating hyperplane in order to develop a hyperplane that almost separates the classes, using a so-called **soft margin**.

- The margin is **soft** because it can be violated by some of the training observations.
- The generalization of the maximal margin classifier to the non-separable case is known as the **support vector classifier**

Support Vector Classifier



- The hyperplane is shown as a solid line and the margins are shown as dashed lines
- Observations 3, 4, 5, and 6 are on the correct side of the margin, 2 is on the margin, and 1 is on the wrong side of the margin
- Observations 7 and 10 are on the correct side of the margin, 9 is on the margin, and 8 is on the wrong side of the margin.

- The support vector classifier classifies a test observation depending on which side of a hyperplane it lies.

- The hyperplane is chosen to correctly separate most of the training observations into the two classes, but may misclassify a few observations

- The support vector classifier selects a hyperplane by solving the problem
- Maximize the margin M over the set of $\beta_0, \beta_1, \cdots, \beta_p, \epsilon_1, \cdots, \epsilon_n, M$ such that

$$\sum_{j=1}^{p} \beta_j^2 = 1 \tag{3}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i) \tag{4}$$

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C \tag{5}$$

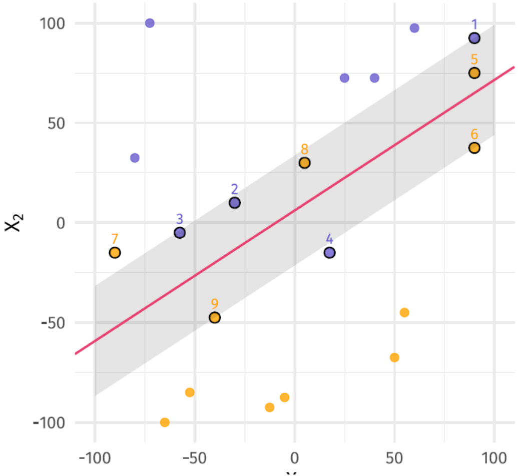
- M is the width of the margin; we seek to maximize this quantity
- ϵ_i are **slack variables** that allow i to violate the margin or hyperplane

- The slack variable ϵ_i tells us where the ith observation is located, relative to the hyperplane and relative to the margin

- If $\epsilon_i = 0$ then the ith observation is on the correct side of the margin

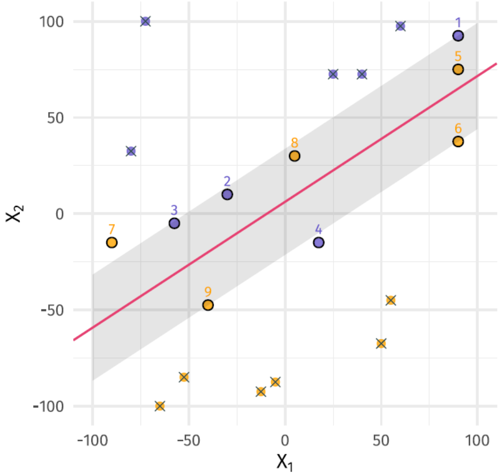
- If $\epsilon_i > 0$ then the ith observation is on the wrong side of the margin, and we say that the ith observation has violated the margin.

- If $\epsilon_i > 1$ then the ith observation is on the wrong side of the hyperplane



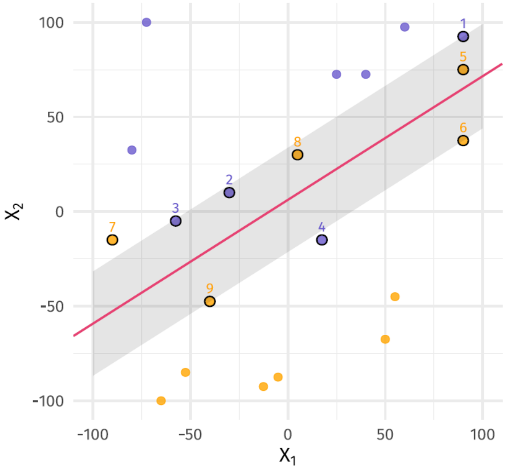
For $\epsilon_i = 0$:

- $M\,(1 - \epsilon_i) > 0$
- Correct side of hyperplane
- Correct side of margin (or on margin)
- No cost (C)
- Distance $\geq M$
- *Examples?*



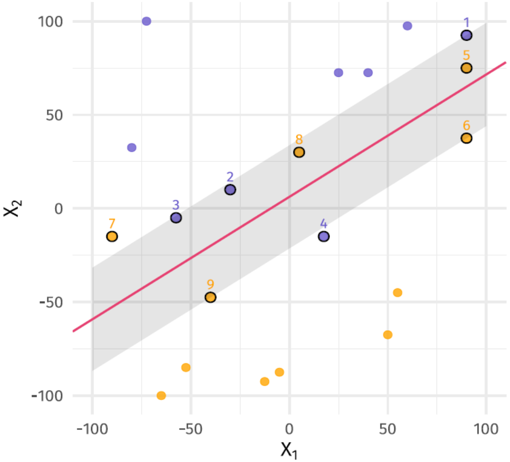
For $\epsilon_i = 0$:

- $M\,(1 - \epsilon_i) > 0$
- Correct side of hyperplane
- Correct side of margin (or on margin)
- No cost (C)
- Distance $\geq M$
- Correct side of margin: $(\times)$
- *On margin*: 1, 6, 9



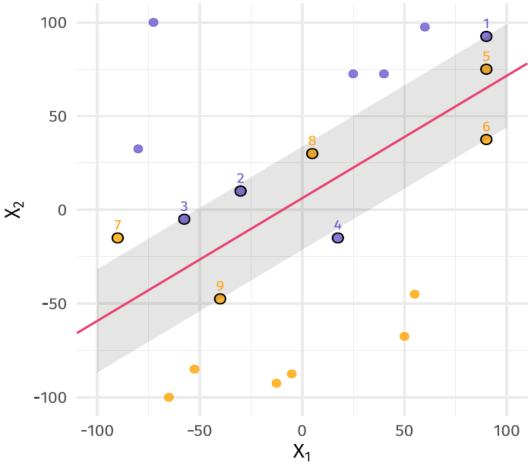
For $0 \leq \epsilon_i \leq 1$:

- $M\left(1 - \epsilon_i\right) > 0$
- Correct side of hyperplane
- Wrong side of the margin (*violates margin*)
- Pays cost ϵ_i
- Distance $< M$
- *Examples?*



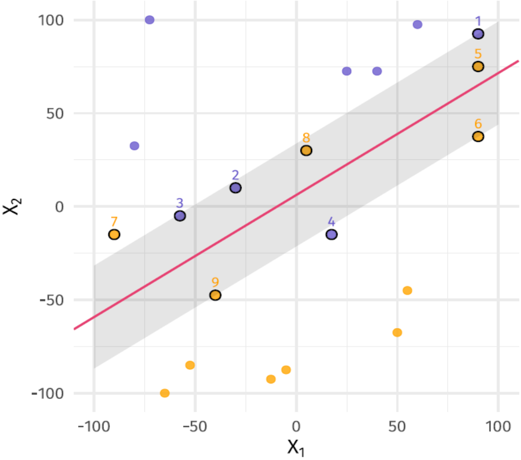
For $0 \leq \epsilon_i \leq 1$:

- $M\,(1 - \epsilon_i) > 0$
- Correct side of hyperplane
- Wrong side of the margin (*violates margin*)
- Pays cost ϵ_i
- Distance $< M$
- *Ex:* 2, 3



For $\epsilon_i \geq 1$:

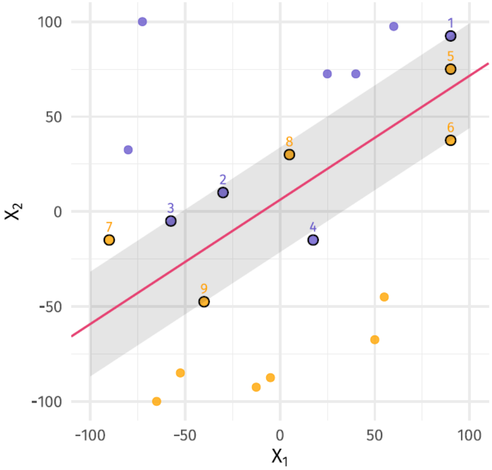
- $M(1 - \epsilon_i) < 0$
- Wrong side of hyperplane
- Pays cost ϵ_i
- Distance $\lesseqqgtr M$
- *Examples?*



For $\epsilon_i \geq 1$:

- $M\left(1-\epsilon_i\right)<0$
- Wrong side of hyperplane
- Pays cost ϵ_i
- Distance $\lesseqqgtr M$
- *Ex:* 4, 5, 7, 8

- It turns out that only observations that either lie on the margin or that violate the margin will affect the hyperplane, and hence the classifier obtained.
- In other words, an observation that lies strictly on the correct side of the margin does not affect the support vector classifier!
- Changing the position of that observation would not change the classifier at all, provided that its position remains on the correct side of the margin.
- Observations that lie directly on the margin, or on the wrong side of the margin for their class, are known as **support vectors**.
- These observations do affect the support vector classifier.



Support vectors

- On margin
- Violate margin
- Wrong side of hyperplane

determine the classifier.

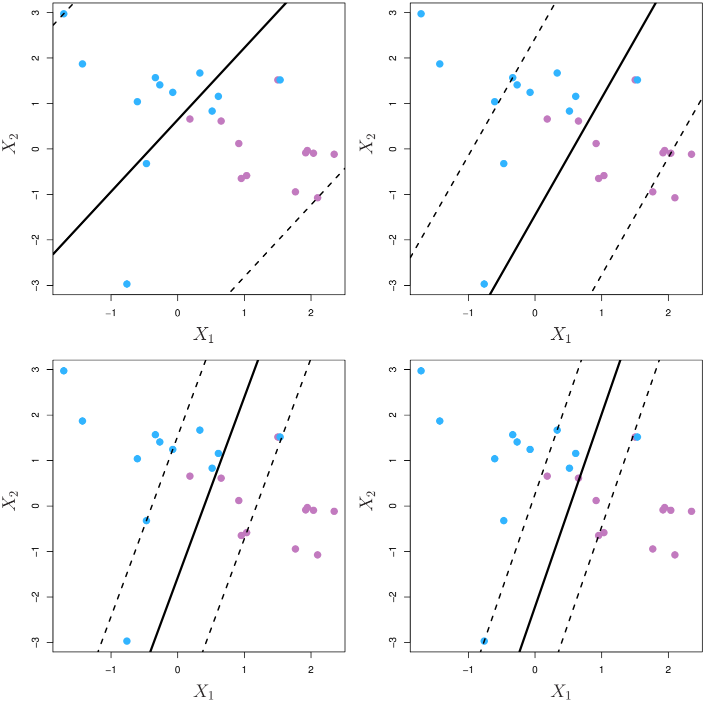
- C bounds the sum of the ϵ_i's, and so it determines the number and severity of the violations to the margin (and to the hyperplane) that we will tolerate.
- We can think of C as a budget for the amount that the margin can be violated by the n observations.
- If $C = 0$ then there is no budget for violations to the margin, and it must be the case that $\epsilon_1 = \cdots = \epsilon_n = 0$, in which case we're back to the maximal margin hyperplane optimization problem

- As the budget C increases, we become more tolerant of violations to the margin, and so the margin will widen.
- Conversely, as C decreases, we become less tolerant of violations to the margin and so the margin narrows.
- In practice, C is treated as a tuning parameter that is generally chosen via cross-validation.

- As with the tuning parameters that we have seen throughout this class, C controls the bias-variance trade-off of the statistical learning technique.

- When C is small, we seek narrow margins that are rarely violated; this amounts to a classifier that is highly fit to the data, which may have low bias but high variance.

- On the other hand, when C is larger, the margin is wider and we allow more violations to it; this amounts to fitting the data less hard and obtaining a classifier that is potentially more biased but may have lower variance.



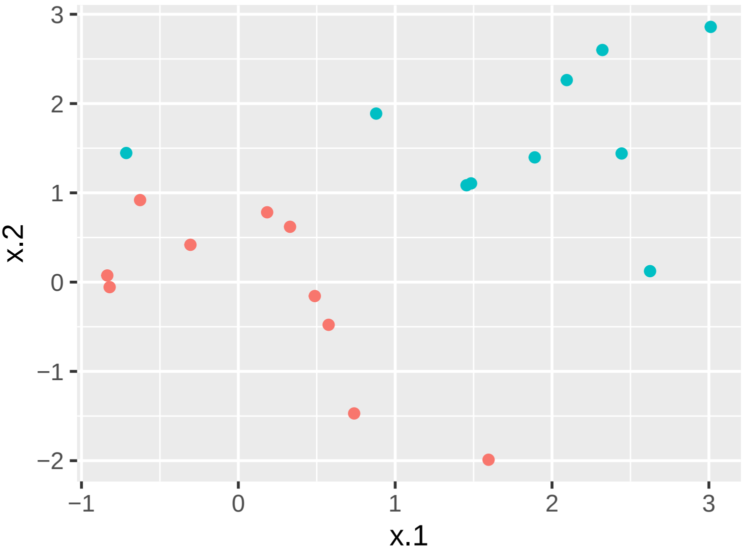
X_1
X_2
−1
0
1
2
−3
−2
−1
0
1
2
3

- The `svm()` function from the `e1071` package can be used to fit a support vector classifier when the argument `kernel="linear` is used
- A cost argument allows us to specify the cost of a violation to the margin. When the cost argument is small, then the margins will be wide and many support vectors will be on the margin or will violate the margin.
- When the cost argument is large, then the margins will be narrow and there will be few support vectors on the margin or violating the margin.

Application

```
x=matrix (rnorm (20*2) , ncol =2)
y=c(rep (-1,10) , rep (1 ,10) )
x[y==1 ,]= x[y==1,] + 1.5
data=data.frame(x=x,y=as.factor (y))
```

- The two classes are linearly separable



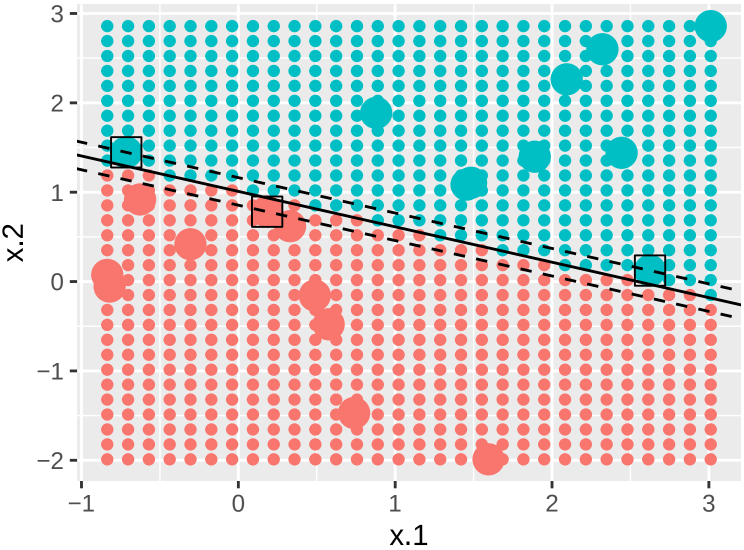
- We fit the support vector classifier and plot the resulting hyperplane, using a very large value of cost so that no observations are misclassified

```
svmfit =svm(y ~ ., data=data , kernel ="linear",
            cost =10000, scale =FALSE)
summary(svmfit)
```

```
Call:
svm(formula = y ~ ., data = data, kernel = "linear", cost =
    scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10000

Number of Support Vectors:  3

 ( 1 2 )
```

```
make.grid = function(x, n = 30){
  grange = apply(x, 2, range)
  x1 = seq(from = grange[1,1], to = grange[2,1], length = n
  x2 = seq(from = grange[1,2], to = grange[2,2], length = n
  expand.grid(x.1 = x1, x.2 = x2)}
xgrid = make.grid(x)
```

- The margin is very narrow. It seems likely that this model will perform poorly on test data.



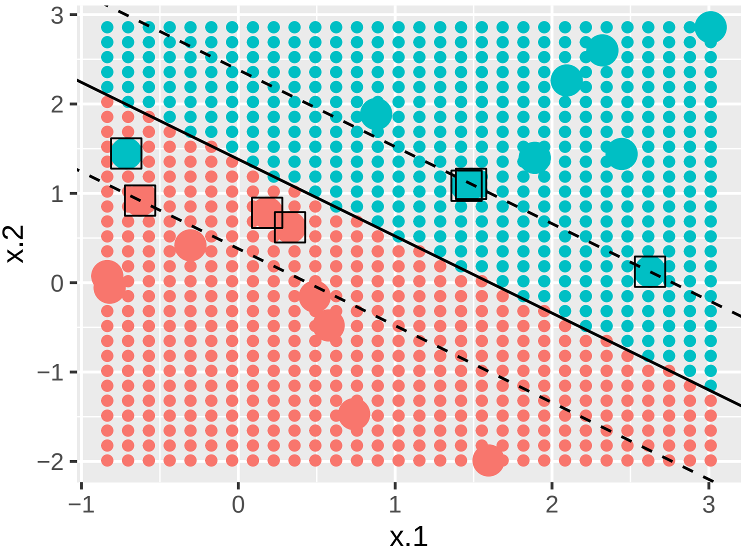
- Let's try a smaller value of cost

```
svmfit =svm(y ~ ., data=data , kernel ="linear",
              cost =1, scale =FALSE)
summary(svmfit)
```

```
Call:
svm(formula = y ~ ., data = data, kernel = "linear", cost =


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  7

 ( 3 4 )


Number of Classes:  2
```

```
make.grid = function(x, n = 30){
  grange = apply(x, 2, range)
  x1 = seq(from = grange[1,1], to = grange[2,1], length = n
  x2 = seq(from = grange[1,2], to = grange[2,2], length = n
  expand.grid(x.1 = x1, x.2 = x2)}
xgrid = make.grid(x)
```

- Using `cost=1`, we misclassify a training observation, but we also obtain a much wider margin and make use of seven support vectors. It seems likely that this model will perform better on test data than the model with larger cost



```
x=matrix (rnorm (20*2) , ncol =2)
y=c(rep (-1,10) , rep (1 ,10) )
x[y==1 ,]= x[y==1,] + 1
data=data.frame(x=x, y=as.factor (y))

xtest=matrix (rnorm (20*2) , ncol =2)
ytest=sample (c(-1,1) , 20, rep=TRUE)
xtest[ytest ==1 ,]= xtest[ytest ==1,] + 1
testdata =data.frame (x=xtest , y=as.factor (ytest))
```

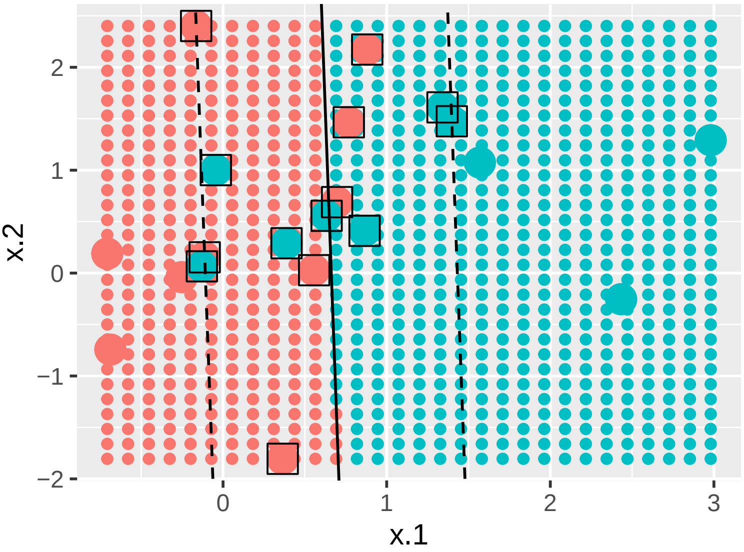
- The two classes are not linearly separable

```
ggplot(data,aes(x=x.1,y=x.2, colour=y))+
  geom_point()+
    theme(legend.position = "None")
```

```r
svmfit =svm(y ~ ., data=data , kernel ="linear",
            cost =10, scale =FALSE)
summary(svmfit)
```

```
Call:
svm(formula = y ~ ., data = data, kernel = "linear", cost =


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10

Number of Support Vectors:  14

 ( 7 7 )


Number of Classes:  2
```

```
make.grid = function(x, n = 30){
  grange = apply(x, 2, range)
  x1 = seq(from = grange[1,1], to = grange[2,1], length = n
  x2 = seq(from = grange[1,2], to = grange[2,2], length = n
  expand.grid(x.1 = x1, x.2 = x2)}
xgrid = make.grid(x)
```

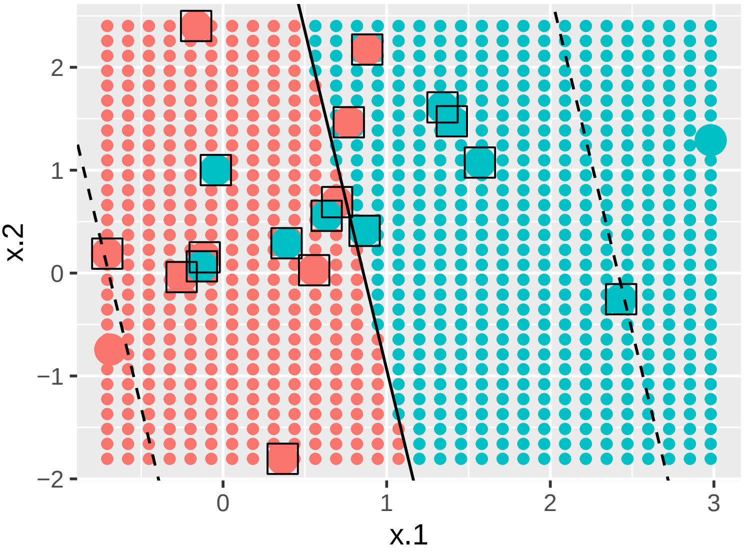


x.2
2
1
0
−1
−2
0
1
2
3
x.1

- Let's use a smaller cost

```
svmfit2 =svm(y ~ ., data=data , kernel ="linear",
             cost =0.1,scale =FALSE )
summary(svmfit2)
```

```
Call:
svm(formula = y ~ ., data = data, kernel = "linear", cost =
    scale = FALSE)



Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.1

Number of Support Vectors:  18

 ( 9 9 )
```



x.2
2
1
0
−1
−2
0
1
2
3
x.1

- The e1071 library includes a built-in function, `tune()`, to perform cross validation.
- By default, `tune()` performs ten-fold cross-validation on a set of models of interest.

```
tune.out = tune(svm ,y~.,data=data ,kernel ="linear",
ranges =list(cost=c(0.001 , 0.01, 0.1, 1,5,10,100) ))
bestmod =tune.out$best.model
bestmod
```

```
Call:
best.tune(method = svm, train.x = y ~ ., data = data, range
    0.01, 0.1, 1, 5, 10, 100)), kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.1
```

- Make predictions on the test data

```r
ypred=predict(bestmod ,testdata )
cm <- confusionMatrix(data=ypred,
                 reference=testdata$y,
                 positive="1")
cm$table
```

```
          Reference
Prediction -1 1
        -1  7 4
        1   4 5
```

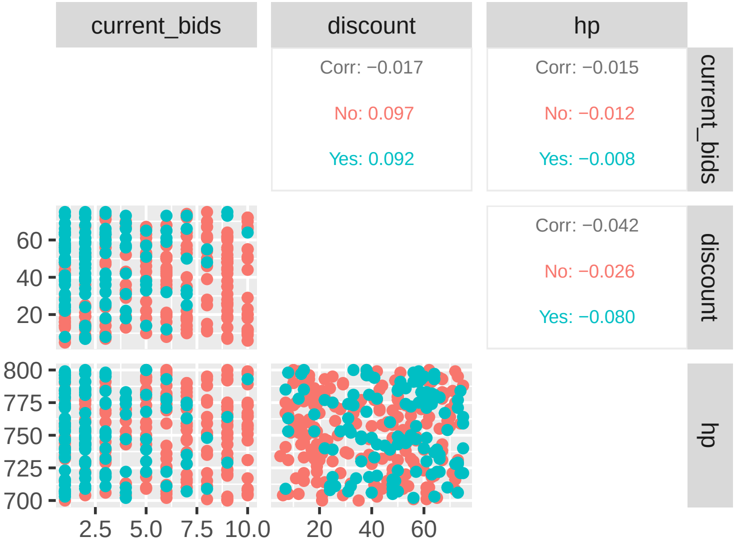
```r
c(cm$overall[1],cm$byClass[c(1,2,7)])
```

```
   Accuracy Sensitivity Specificity          F1
      0.600       0.556       0.636       0.556
```

Exercise

```r
cars_train <- read.csv("cayugacars_train.csv")
cars_test <- read.csv("cayugacars_test.csv")
cars_train <- select(cars_train,-c(10:ncol(cars_train)))
cars_test <- select(cars_test,-c(10:ncol(cars_test)))
```

- Create pairwise correlations



- Tune the svm model

```
tune.cars = tune(svm ,customer_bid~.,data=cars_train ,kerne
ranges =list(cost=c(0.001,0.01,0.05,0.1,1,10,100) ))
bestmod =tune.cars$best.model
bestmod
```

```
Call:
best.tune(method = svm, train.x = customer_bid ~ ., data =
    ranges = list(cost = c(0.001, 0.01, 0.05, 0.1, 1, 10, 1
    kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.05

Number of Support Vectors:  673
```

- Make predictions on the test data and call the confusion matrix

```r
ypred=predict(bestmod , cars_test )
cm <- confusionMatrix(data=ypred,
                reference=cars_test$customer_bid,
                positive="Yes")
cm$table
```

```
          Reference
Prediction  No Yes
       No  160  37
       Yes  28  82
```

```r
c(cm$overall[1],cm$byClass[c(1,2,7)])
```

```
   Accuracy Sensitivity Specificity          F1
      0.788       0.689       0.851       0.716
```