25) Maximal Margin Classifier, and Support Vector Classifiers

Vitor Kamada

March 2018

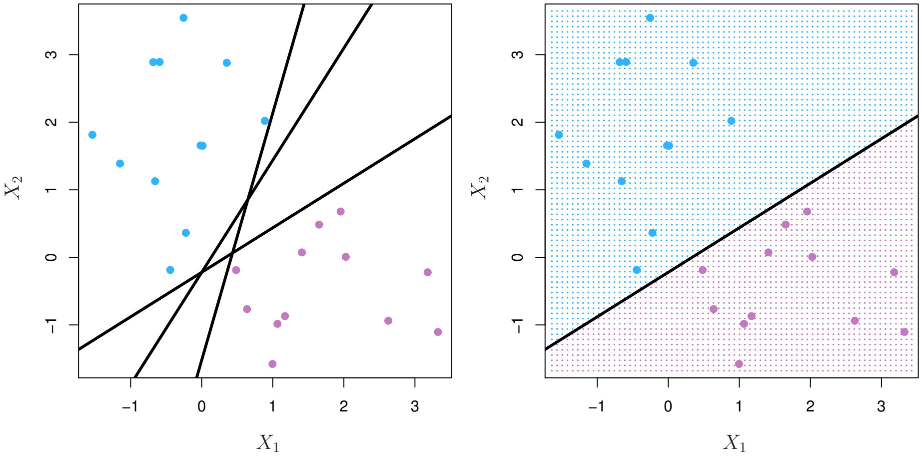
Reference

Tables, Graphics, and Figures from

An Introduction to Statistical Learning

James et al. (2017): Chapters: 9.1, 9.2, and 9.6.1

Separating Hyperplane



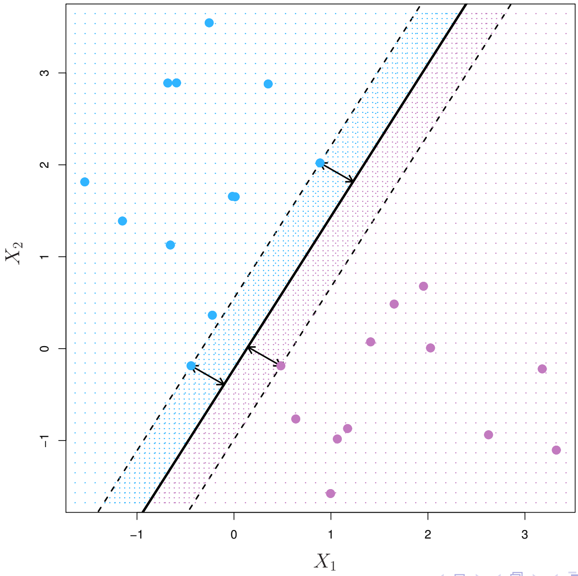
Hyperplane

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} > 0 \text{ if } y_i = 1$$

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} < 0 \text{ if } y_i = -1$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip}) > 0$$

Maximal Margin Classifier



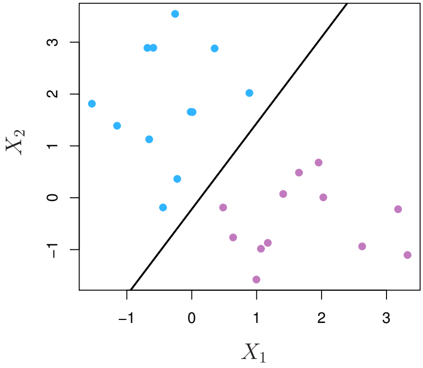
Construction of the Maximal Margin Classifier

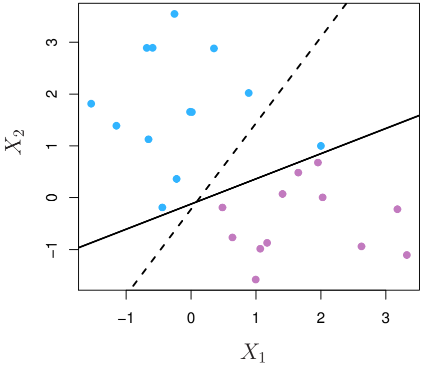
$$\underset{\beta_0,\beta_1,\ldots,\beta_p,M}{maximize} \quad M$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1$$

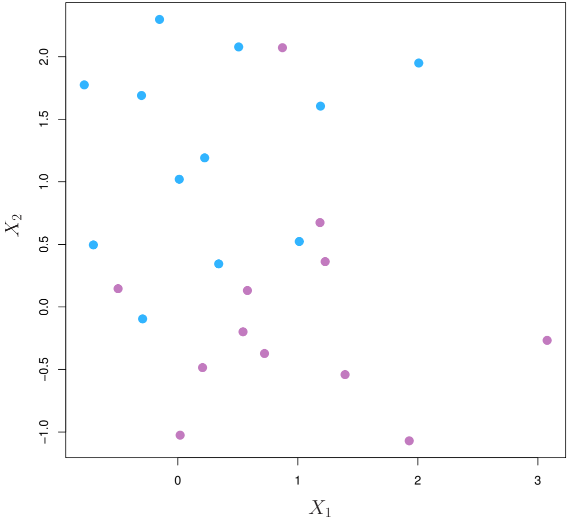
$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M$$

Maximal Margin Hyperplane

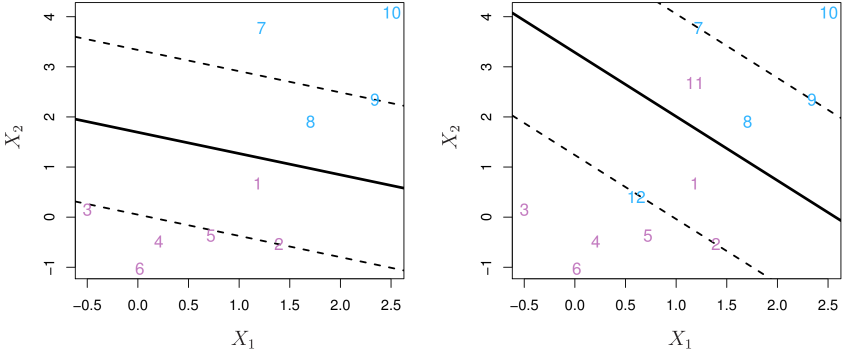




No Separating Hyperplane



Soft Margin Classifier



Construction of Support Vector Classifier

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{maximize} \quad M$$

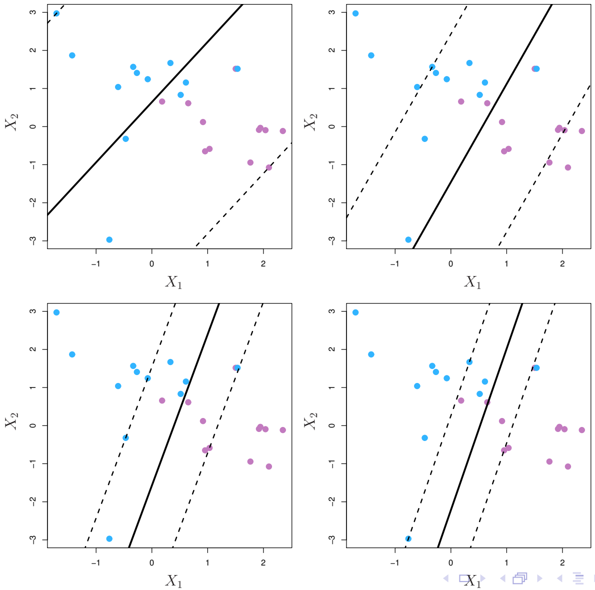
$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1$$

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

$$\epsilon_i \geq 0$$

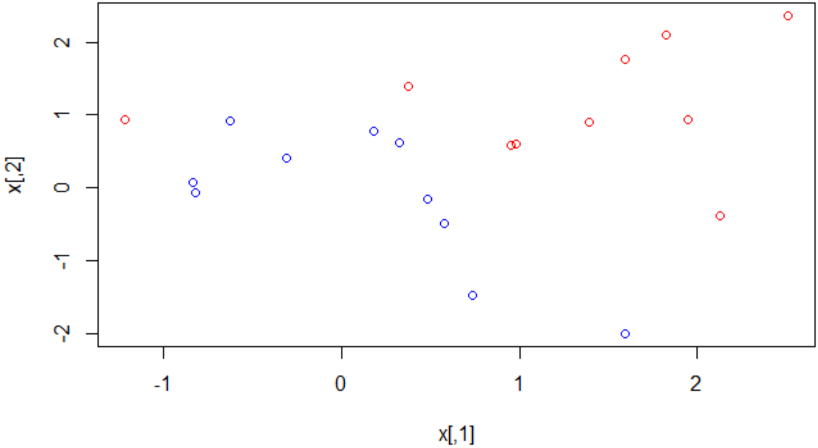
$$\sum_{i=1}^{n} \epsilon_i \leq C$$

Different Values of the Tuning Parameter C



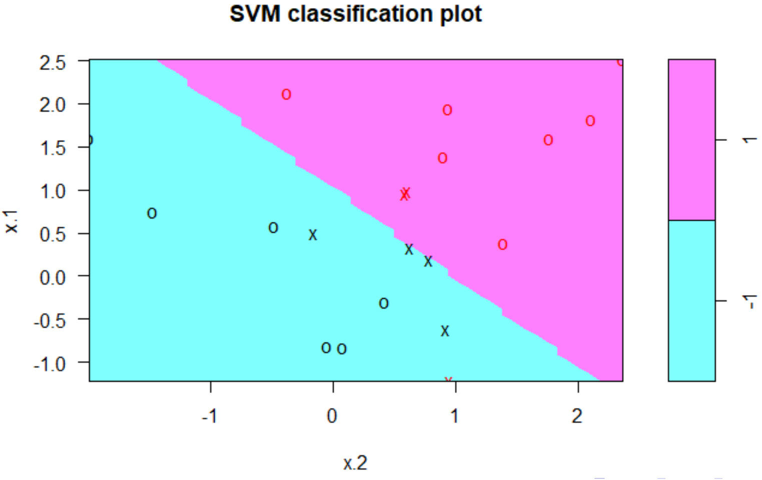
set.seed(1); x=matrix(rnorm(20*2), ncol=2)

```
y=c(rep(-1,10), rep(1,10)); x[y==1,]=x[y==1,] + 1
plot(x, col=(3-y))
```

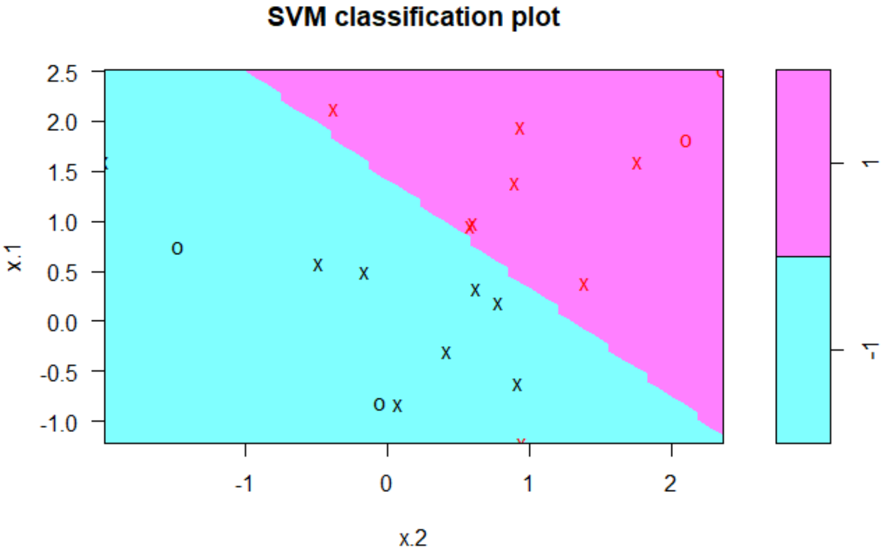


dat=data.frame(x=x,y=as.factor(y)); library(e1071)

svmfit=svm(y~., data=dat, kernel="linear", cost=10,scale=FALSE); plot(svmfit, dat)



svmfit=svm(y∼., data=dat, kernel="linear", cost=0.1,scale=FALSE); plot(svmfit, dat)



set.seed(1)

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

summary(tune.out)

```
- sampling method: 10-fold cross validation

- best parameters:
 cost
  0.1

- best performance: 0.1

- Detailed performance results:
   cost error dispersion
1 1e-03  0.70  0.4216370
2 1e-02  0.70  0.4216370
3 1e-01  0.10  0.2108185
4 1e+00  0.15  0.2415229
5 5e+00  0.15  0.2415229
6 1e+01  0.15  0.2415229
7 1e+02  0.15  0.2415229
```

bestmod=tune.out$best.model

```
xtest=matrix(rnorm(20*2), ncol=2)
ytest=sample(c(-1,1), 20, rep=TRUE)
xtest[ytest==1,]=xtest[ytest==1,] + 1
testdat=data.frame(x=xtest, y=as.factor(ytest))
ypred=predict(bestmod,testdat)
table(predict=ypred, truth=testdat$y)
```

```
        truth
predict -1  1
     -1 11  1
     1   0  8
```

svmfit=svm(y∼., data=dat, kernel="linear", cost=.01,scale=FALSE)

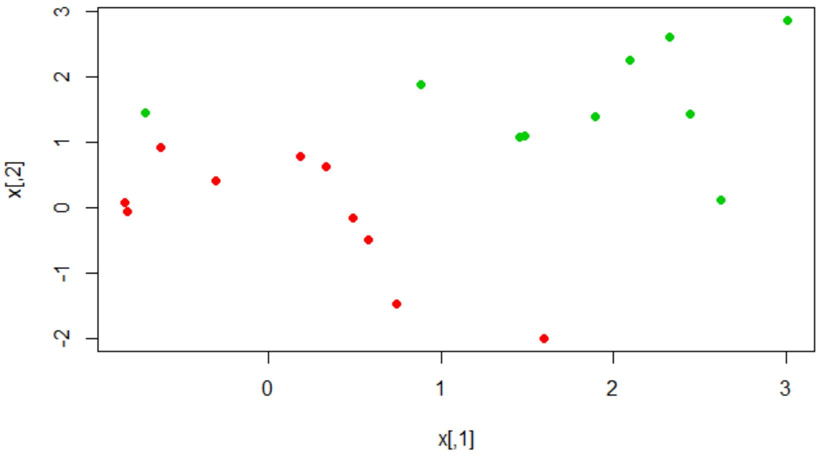
ypred=predict(svmfit,testdat)

table(predict=ypred, truth=testdat$y)

```
         truth
predict -1  1
     -1 11  2
     1   0  7
```

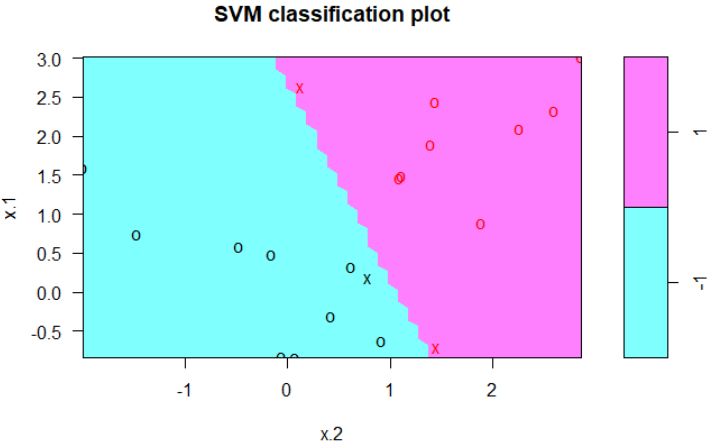
x[y==1,]=x[y==1,]+0.5

plot(x, col=(y+5)/2, pch=19)



dat=data.frame(x=x,y=as.factor(y))

svmfit=svm(y~., data=dat, kernel="linear", cost=1e5); plot(svmfit, dat)



svmfit=svm(y∼.,data=dat,kernel="linear", cost=1)

plot(svmfit,dat)