26) Support Vector Machines (SVM)

Vitor Kamada

March 2018

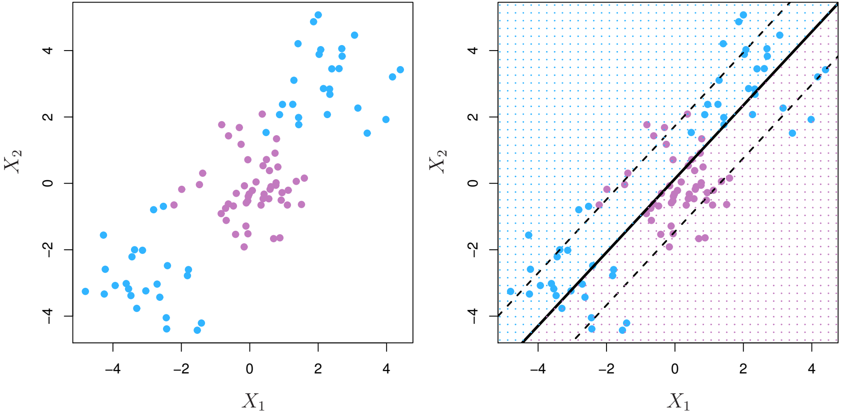
Reference

Tables, Graphics, and Figures from

An Introduction to Statistical Learning

James et al. (2017): Chapters: 9.3, 9.4, 9.6.2, 9.6.3, 9.6.4

Non-Linear Boundary



Support Vector Machines

$$\underset{\beta_0,\beta_{11},\beta_{12}\ldots,\beta_{p1},\beta_{p2},\epsilon_1,\ldots,\epsilon_n,M}{maximize} \quad M$$

$$\text{subject to } \sum_{j=1}^{p}\sum_{k=1}^{2}\beta_{jk}^2 = 1$$

$$y_i(\beta_0 + \sum_{j=1}^{p}\beta_{j1}x_{ij} + \sum_{j=1}^{p}\beta_{j2}x_{ij}^2) \geq M(1-\epsilon_i)$$

$$\epsilon_i \geq 0$$

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

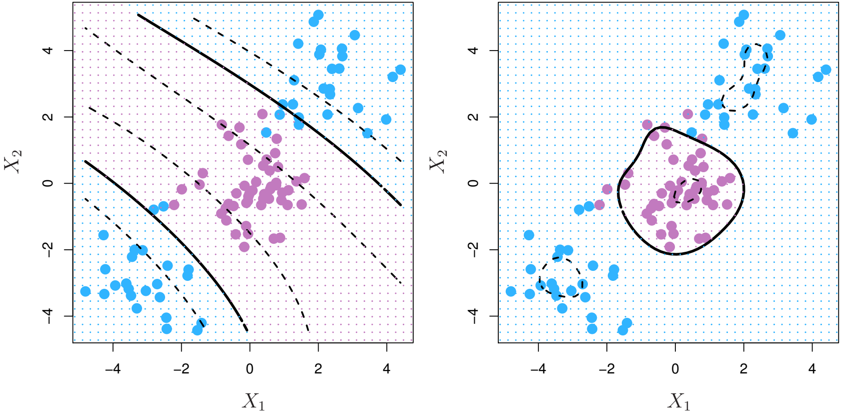
$Kernel = K(x_i, x_{i'})$

Linear: $\sum_{j=1}^{p} (x_{ij}, x_{i'j})$

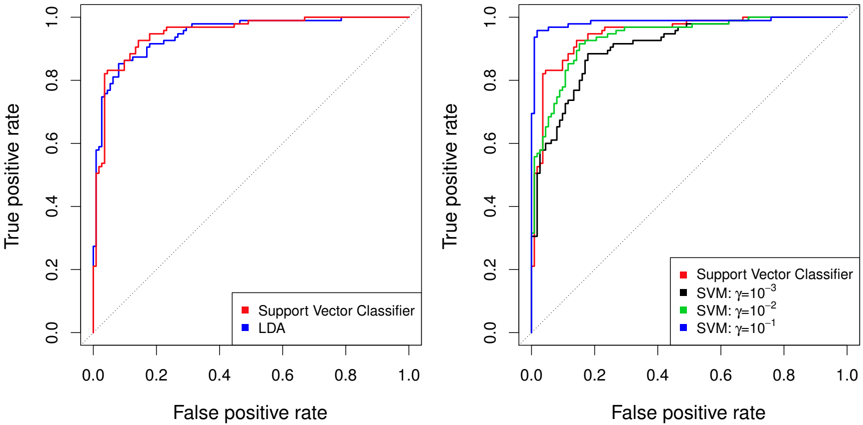
Polynomial: $(1 + \sum_{j=1}^{p} x_{ij}, x_{i'j})^d$

Radial: $exp[-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2]$

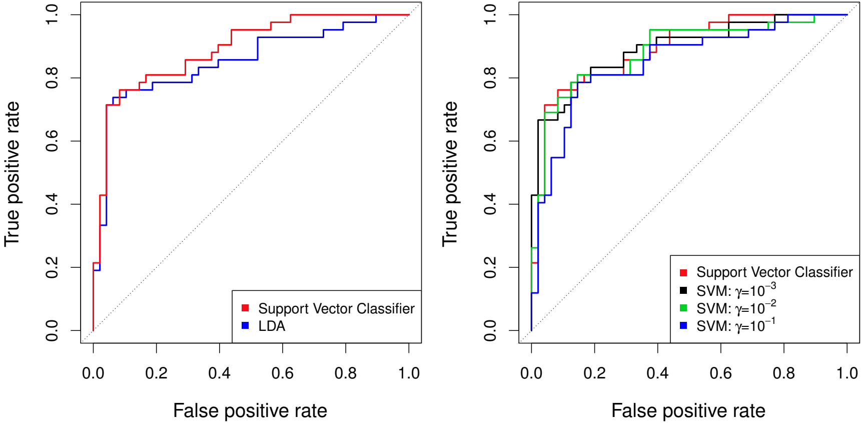
Polynomial Kernel of Degree 3 vs Radial Kernel



Heart Data Training Set



Heart Data Test Set

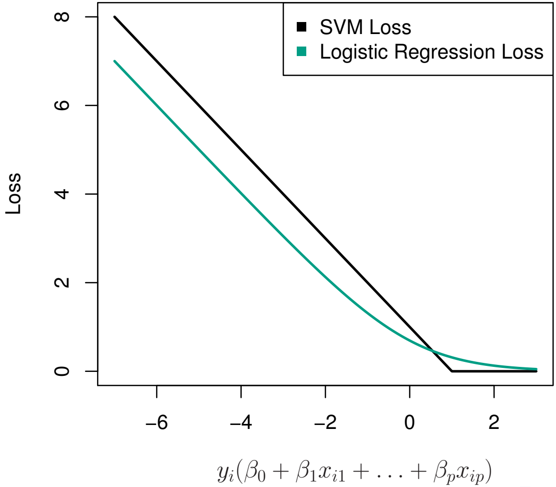


Support Vector Machine Loss Function

$$f(X) = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p$$

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{minimize} \{ \sum_{i=1}^{n} max[0, 1 - y_i f(x_i)] + \lambda \sum_{j=1}^{p} \beta_j^2 \}$$

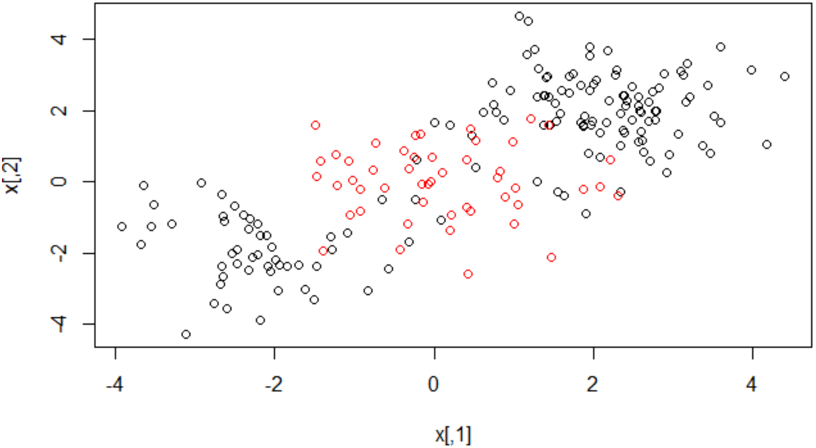
SVM vs Logistic Regression Loss Function



library(e1071); set.seed(1)

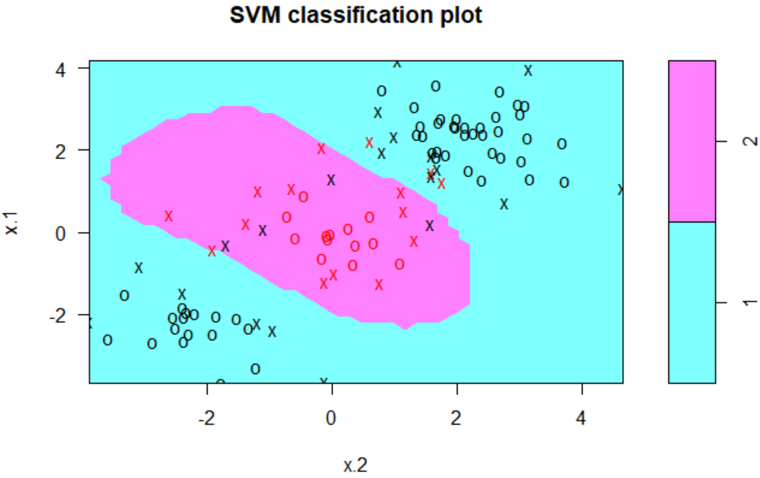
```
x=matrix(rnorm(200*2), ncol=2)
x[1:100,]=x[1:100,]+2
x[101:150,]=x[101:150,]-2
y=c(rep(1,150),rep(2,50))
dat=data.frame(x=x,y=as.factor(y))
```

plot(x, col=y)

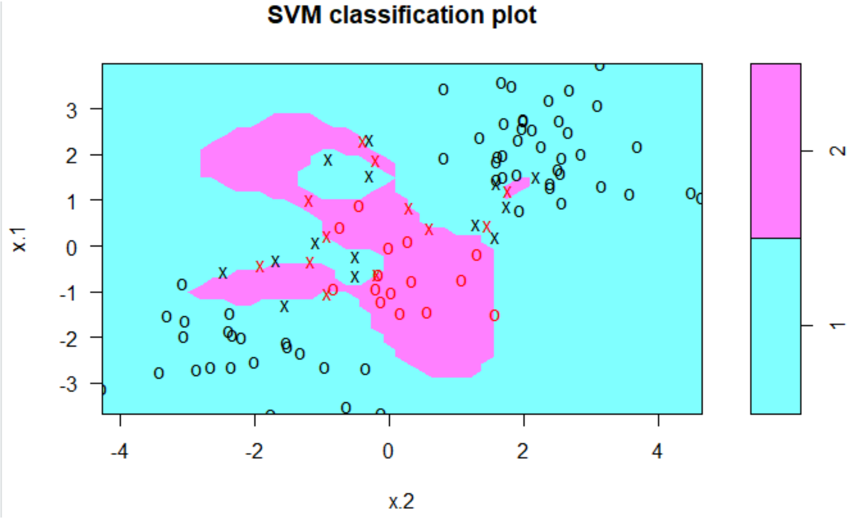


train=sample(200,100)

svmfit=svm(y∼., data=dat[train,], kernel="radial", gamma=1, cost=1); plot(svmfit, dat[train,])

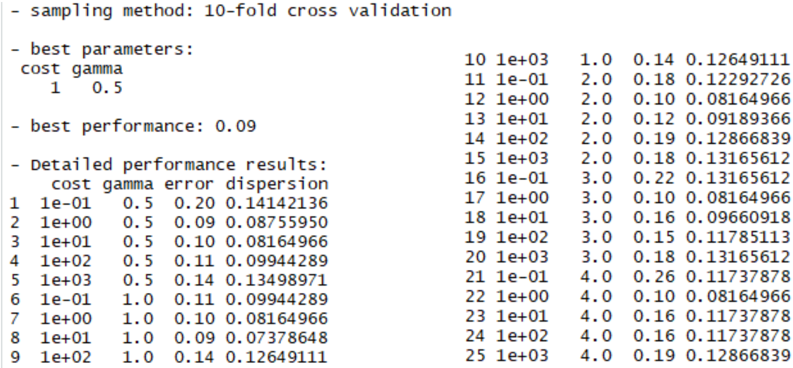


svmfit=svm(y∼., data=dat[train,], kernel="radial",gamma=1,cost=1e5)



tune.out=tune(svm, y∼., data=dat[train,], kernel="radial", ranges=list(cost=c(0.1,1,10,100,1000),gamma=c(0.5,1,2,3,4)))

summary(tune.out)



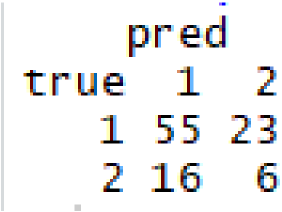
```
- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
    1   0.5

- best performance: 0.09

- Detailed performance results:
    cost gamma error dispersion
1  1e-01   0.5  0.20 0.14142136
2  1e+00   0.5  0.09 0.08755950
3  1e+01   0.5  0.10 0.08164966
4  1e+02   0.5  0.11 0.09944289
5  1e+03   0.5  0.14 0.13498971
6  1e-01   1.0  0.11 0.09944289
7  1e+00   1.0  0.10 0.08164966
8  1e+01   1.0  0.09 0.07378648
9  1e+02   1.0  0.14 0.12649111
10 1e+03   1.0  0.14 0.12649111
11 1e-01   2.0  0.18 0.12292726
12 1e+00   2.0  0.10 0.08164966
13 1e+01   2.0  0.12 0.09189366
14 1e+02   2.0  0.19 0.12866839
15 1e+03   2.0  0.18 0.13165612
16 1e-01   3.0  0.22 0.13165612
17 1e+00   3.0  0.10 0.08164966
18 1e+01   3.0  0.16 0.09660918
19 1e+02   3.0  0.15 0.11785113
20 1e+03   3.0  0.18 0.13165612
21 1e-01   4.0  0.26 0.11737878
22 1e+00   4.0  0.10 0.08164966
23 1e+01   4.0  0.16 0.11737878
24 1e+02   4.0  0.16 0.11737878
25 1e+03   4.0  0.19 0.12866839
```

table(true=dat[-train,"y"], pred= predict(tune.out$best.model,newx=dat[-train,]))



```
     pred
true  1  2
   1 55 23
   2 16  6
```

library(ISLR); dim(Khan$xtrain)

63 2308

dim(Khan$xtest)

20 2308

table(Khan$ytrain)

1	2	3	4
8	23	12	20

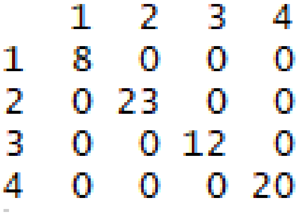
table(Khan$ytest)

1	2	3	4
3	6	6	5

dat=data.frame(x=Khan$xtrain, y=as.factor(Khan$ytrain))

out=svm(y∼., data=dat, kernel="linear",cost=10)

table(out$fitted, dat$y)

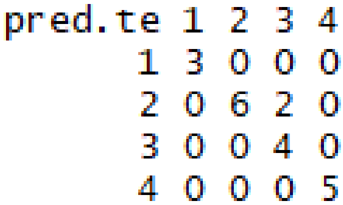


```
     1  2  3  4
  1  8  0  0  0
  2  0 23  0  0
  3  0  0 12  0
  4  0  0  0 20
```

dat.te=data.frame(x=Khan$xtest, y=as.factor(Khan$ytest))

pred.te=predict(out, newdata=dat.te)

table(pred.te, dat.te$y)



```
pred.te 1 2 3 4
      1 3 0 0 0
      2 0 6 2 0
      3 0 0 4 0
      4 0 0 0 5
```