Module 9: Recommended Exercises

TMA4268 Statistical Learning V2020

Stefanie Muff, Department of Mathematical Sciences, NTNU

March xx, 2020

Contents

Recommended exercises

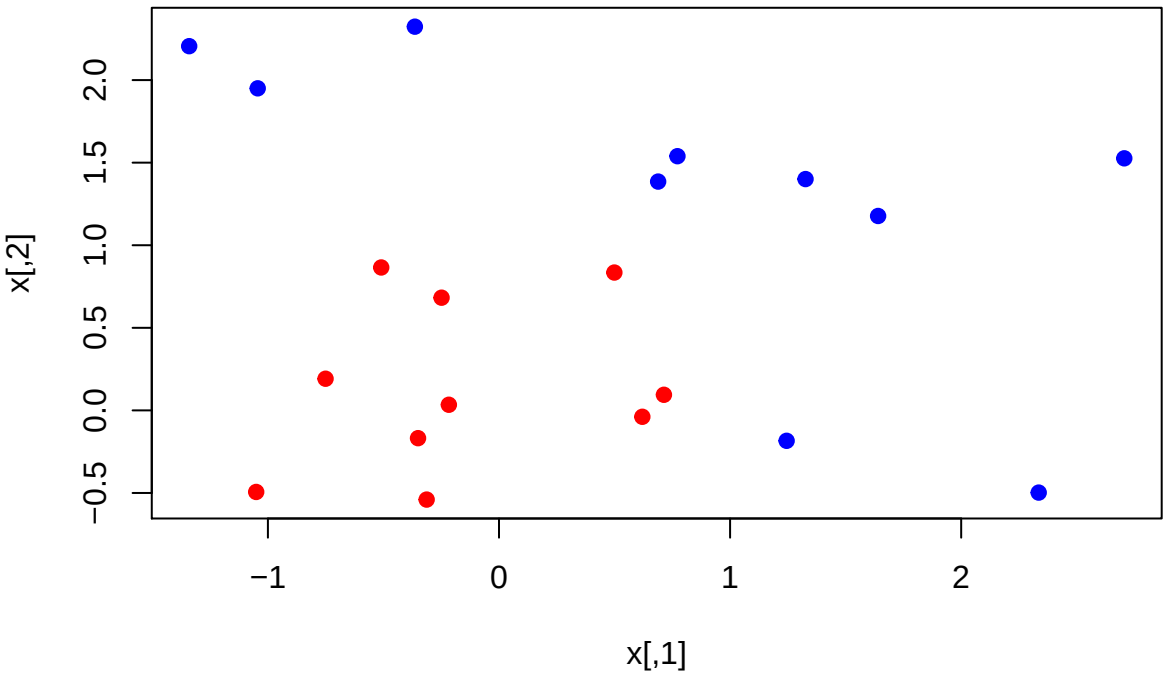
1. Understanding the algorithms:

- Exercise 1, 2 and 3 in the book.

2. Data analysis

- Go back and read in the `forest1` data (is located in the same place as `forest2`) and run the `svm` with a very high value for `cost`. The `forest1` is a separable problem.
- Linear version of SVM: Making nicer plots for SVM from Lab video. Go through the code and see what is happening (and see the video if you want more explanation).

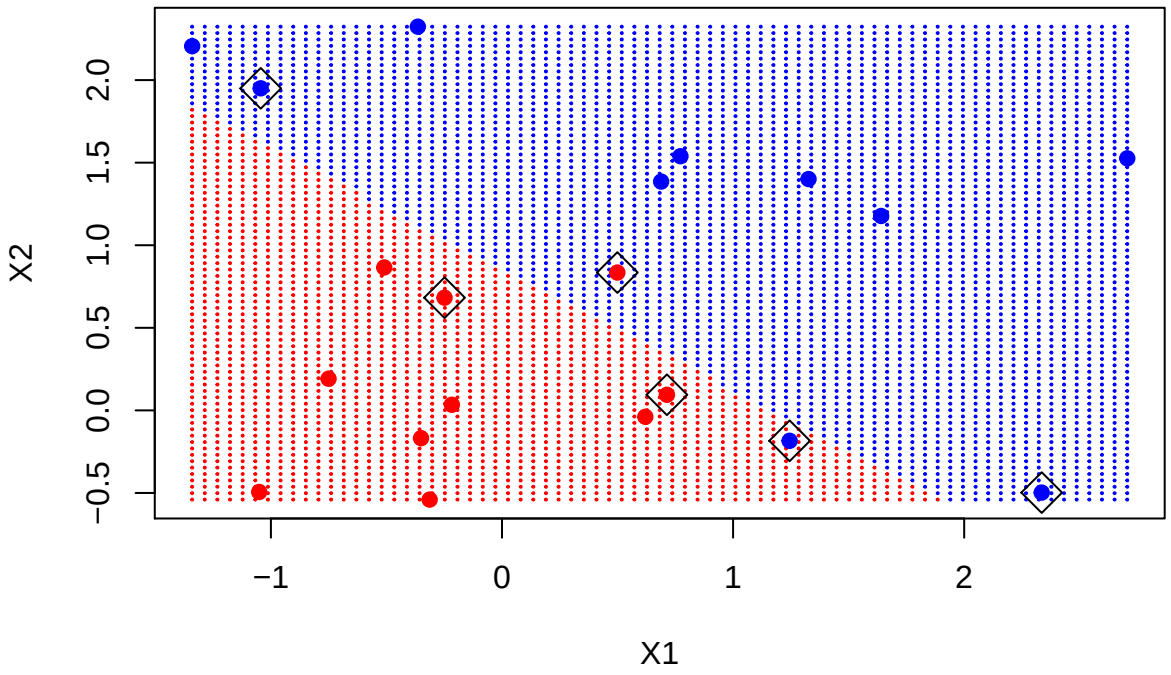
```
# code taken from video by Trevor Hastie linked above
library(e1071)
# fake data
set.seed(10111)
x = matrix(rnorm(40), 20, 2)
y = rep(c(-1, 1), c(10, 10))
x[y == 1, ] = x[y == 1, ] + 1
plot(x, col = y + 3, pch = 19)
```



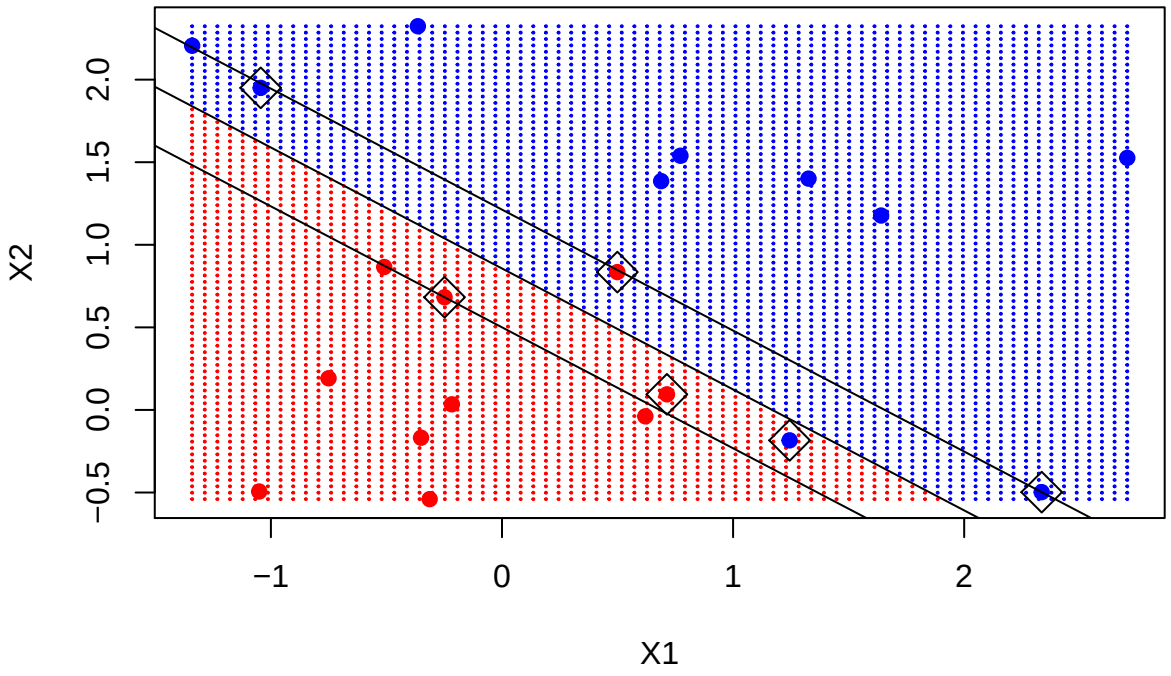

```
# calling svm
dat = data.frame(x, y = as.factor(y))
svmfit = svm(y ~ ., data = dat, kernel = "linear", cost = 10, scale = FALSE)
print(svmfit)
```

```
##
## Call:
## svm(formula = y ~ ., data = dat, kernel = "linear", cost = 10,
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##
## Number of Support Vectors:  6
```

```
# grid for plotting
make.grid = function(x, n = 75) {
    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(X1 = x1, X2 = x2)
}
xgrid = make.grid(x)
ygrid = predict(svmfit, xgrid)
plot(xgrid, col = c("red", "blue")[as.numeric(ygrid)], pch = 20, cex = 0.2)
points(x, col = y + 3, pch = 19)
points(x[svmfit$index, ], pch = 5, cex = 2)
```




```
# more info on results - class boundary

beta = drop(t(svmfit$coefs) %*% x[svmfit$index, ])
beta0 = svmfit$rho
plot(xgrid, col = c("red", "blue")[as.numeric(ygrid)], pch = 20, cex = 0.2)
points(x, col = y + 3, pch = 19)
points(x[svmfit$index, ], pch = 5, cex = 2)
abline(beta0/beta[2], -beta[1]/beta[2])  #class boundary
abline((beta0 - 1)/beta[2], -beta[1]/beta[2])  #class boundary-margin
abline((beta0 + 1)/beta[2], -beta[1]/beta[2])  #class boundary+margin
```

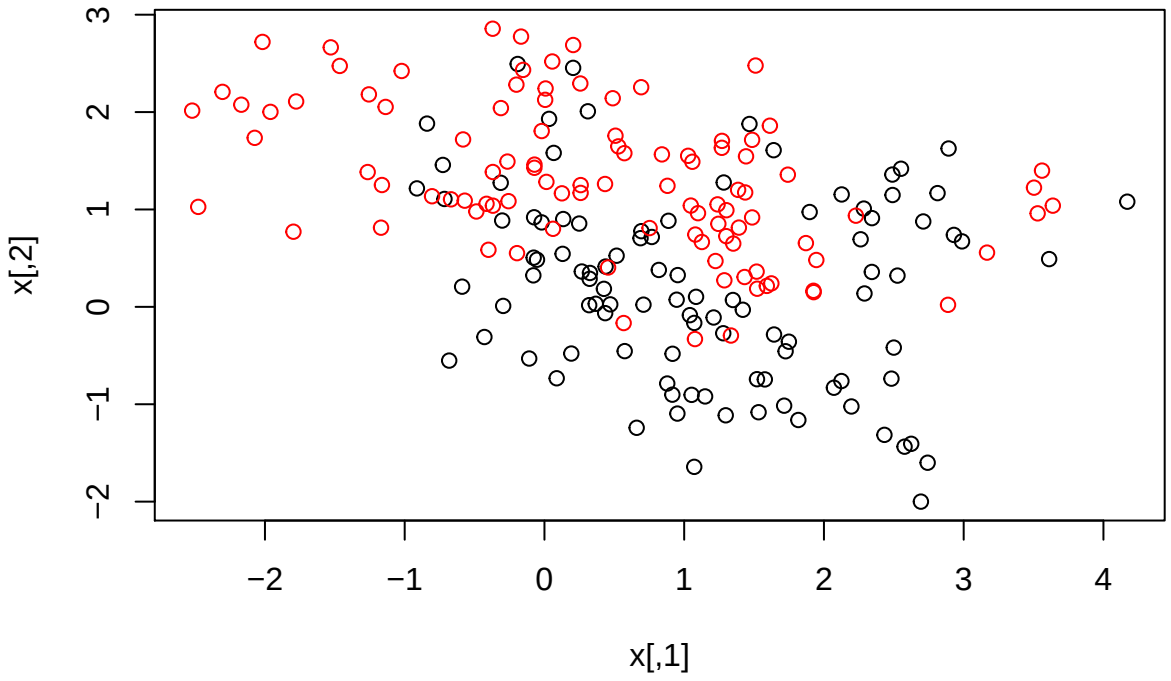


- SVM for non-linear class boundary using simulated data set from @ESL where the truth is known (mixtures of normals probably).

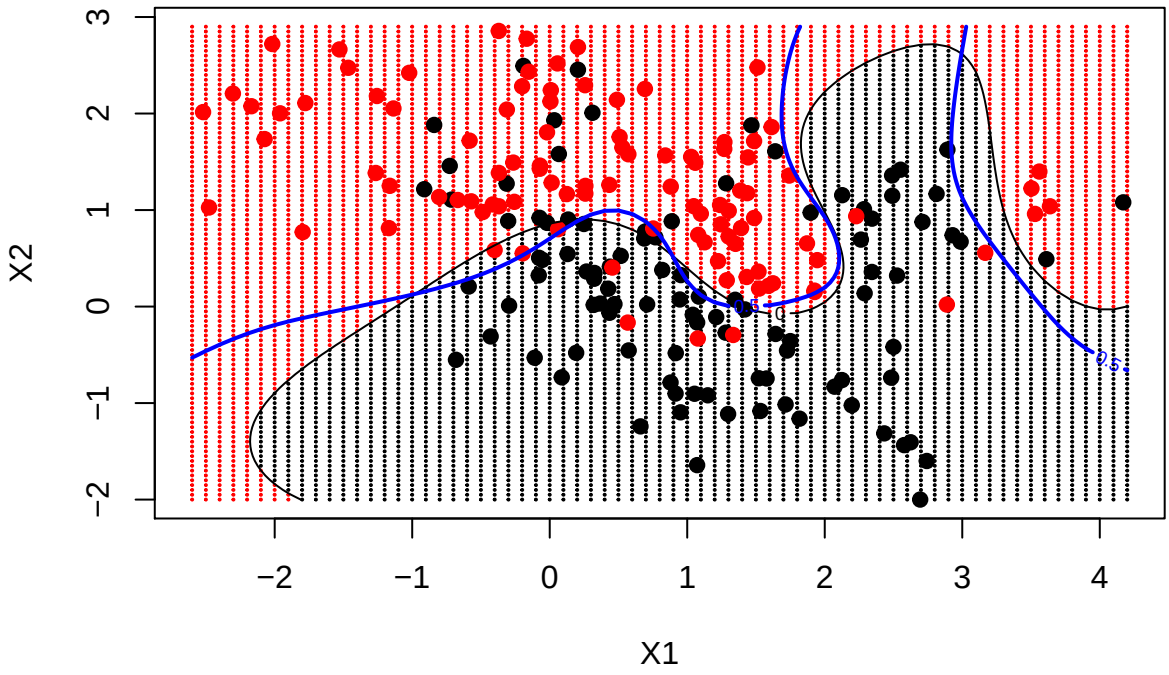
```
load(url("https://web.stanford.edu/~hastie/ElemStatLearn/datasets/ESL.mixture.rda"))
names(ESL.mixture)
```

```
## [1] "x"        "y"        "xnew"     "prob"     "marginal" "px1"
## [7] "px2"      "means"
```

```
rm(x, y)
attach(ESL.mixture)

plot(x, col = y + 1)
```




```
dat = data.frame(y = factor(y), x)
fit = svm(factor(y) ~ ., data = dat, scale = FALSE, kernel = "radial",
    cost = 5)

xgrid = expand.grid(X1 = px1, X2 = px2)
ygrid = predict(fit, xgrid)
plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = 0.2)
points(x, col = y + 1, pch = 19)

# decision boundary
func = predict(fit, xgrid, decision.values = TRUE)
func = attributes(func)$decision
contour(px1, px2, matrix(func, 69, 99), level = 0, add = TRUE)  #svm boundary
contour(px1, px2, matrix(prob, 69, 99), level = 0.5, add = TRUE, col = "blue",
    lwd = 2)  #truth
```

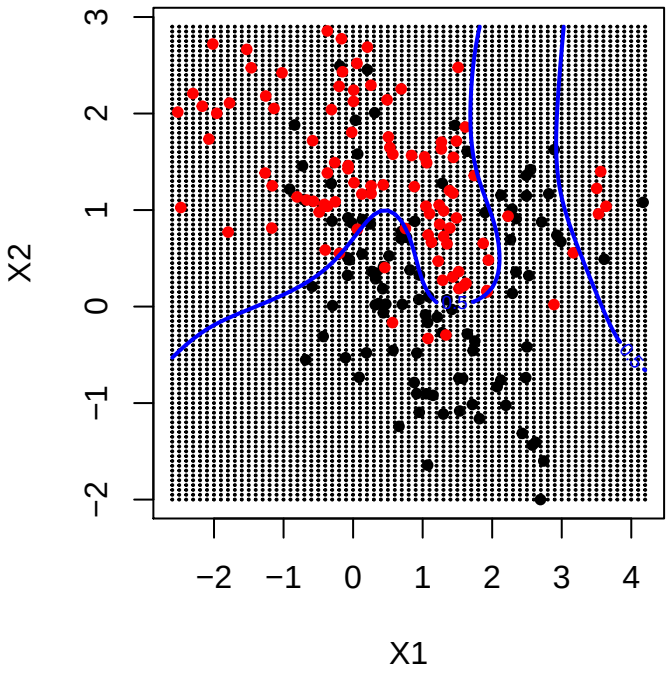


Compulsory exercise 3 2018: Problem 2 - Nonlinear class boundaries and support vector machine

3 a) Bayes decision boundary [1 point]

We will study classification applied to a simulated data set with two classes from @ESL, where the data set is supplied together with the Bayes decision boundary. The boundary is plotted below, together with a training set.

```
load(url("https://web.stanford.edu/~hastie/ElemStatLearn/datasets/ESL.mixture.rda"))
# names(ESL.mixture) prob gives probabilites for each class when the
# true density functions are known px1 and px2 are coordinates in x1
# (length 69) and x2 (length 99) where class probabilites are
# calculated
rm(x, y)
attach(ESL.mixture)
dat = data.frame(y = factor(y), x)
xgrid = expand.grid(X1 = px1, X2 = px2)
par(pty = "s")
plot(xgrid, pch = 20, cex = 0.2)
points(x, col = y + 1, pch = 20)
contour(px1, px2, matrix(prob, 69, 99), level = 0.5, add = TRUE, col = "blue",
    lwd = 2)  #optimal boundary
```



- Q11. What is a Bayes classifier, Bayes decision boundary and Bayes error rate? Hint: pages 37-39 in @ISL.
- Q12. When the Bayes decision boundary is known, do we then need a test set?

b) Support vector machine [1 point]

```
library(e1071)
# support vector classifier
svcfits = tune(svm, factor(y) ~ ., data = dat, scale = FALSE, kernel = "linear",
    ranges = list(cost = c(0.01, 0.1, 1, 5, 10)))
summary(svcfits)
```

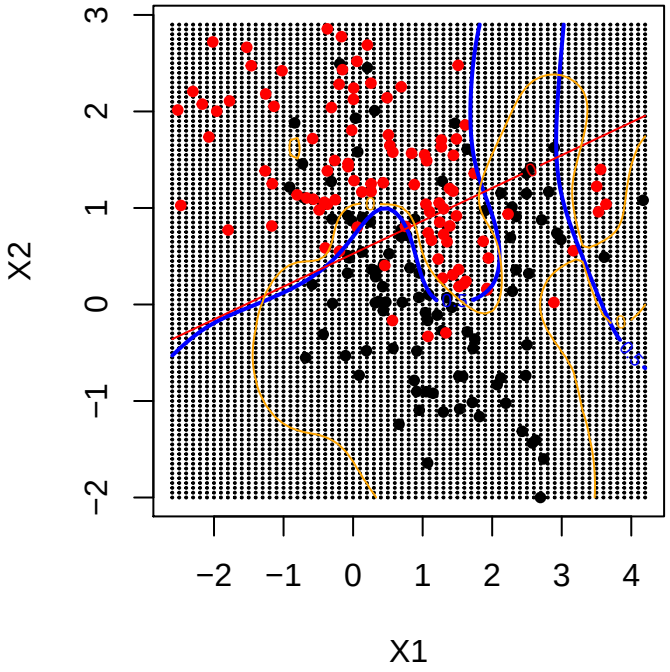
```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##  0.01
##
## - best performance: 0.27
##
## - Detailed performance results:
##    cost error dispersion
## 1  0.01 0.270 0.11595018
## 2  0.10 0.290 0.07745967
## 3  1.00 0.295 0.07619420
## 4  5.00 0.295 0.07619420
## 5 10.00 0.300 0.07817360
```

```r
svcfit = svm(factor(y) ~ ., data = dat, scale = FALSE, kernel = "linear",
    cost = 0.01)
# support vector machine with radial kernel
set.seed(4268)
svmfits = tune(svm, factor(y) ~ ., data = dat, scale = FALSE, kernel = "radial",
    ranges = list(cost = c(0.01, 0.1, 1, 5, 10), gamma = c(0.01, 1, 5,
        10)))
summary(svmfits)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1     5
##
## - best performance: 0.16
##
## - Detailed performance results:
##     cost gamma error dispersion
## 1   0.01  0.01 0.550 0.11547005
## 2   0.10  0.01 0.530 0.12064641
## 3   1.00  0.01 0.275 0.09204468
## 4   5.00  0.01 0.275 0.06346478
## 5  10.00  0.01 0.290 0.08432740
## 6   0.01  1.00 0.535 0.14539219
## 7   0.10  1.00 0.230 0.08232726
## 8   1.00  1.00 0.175 0.06770032
## 9   5.00  1.00 0.175 0.06770032
## 10 10.00  1.00 0.180 0.08881942
## 11  0.01  5.00 0.505 0.20608790
## 12  0.10  5.00 0.400 0.18708287
## 13  1.00  5.00 0.160 0.07745967
## 14  5.00  5.00 0.215 0.09143911
## 15 10.00  5.00 0.195 0.09559754
## 16  0.01 10.00 0.515 0.18566697
## 17  0.10 10.00 0.505 0.18173546
## 18  1.00 10.00 0.190 0.07745967
## 19  5.00 10.00 0.235 0.12483322
## 20 10.00 10.00 0.255 0.13006409
```

```r
svmfit = svm(factor(y) ~ ., data = dat, scale = FALSE, kernel = "radial",
    cost = 1, gamma = 5)

# the same as in a - the Bayes boundary
par(pty = "s")
plot(xgrid, pch = 20, cex = 0.2)
points(x, col = y + 1, pch = 20)
contour(px1, px2, matrix(prob, 69, 99), level = 0.5, add = TRUE, col = "blue",
    lwd = 2)  #optimal boundary

# decision boundaries from svc and svm added
```

```
svcfunc = predict(svcfit, xgrid, decision.values = TRUE)
svcfunc = attributes(svcfunc)$decision
contour(px1, px2, matrix(svcfunc, 69, 99), level = 0, add = TRUE, col = "red")  #svc boundary
svmfunc = predict(svmfit, xgrid, decision.values = TRUE)
svmfunc = attributes(svmfunc)$decision
contour(px1, px2, matrix(svmfunc, 69, 99), level = 0, add = TRUE, col = "orange")  #svm boundary
```



- Q13. What is the difference between a support vector classifier and a support vector machine?
- Q14. What are parameters for the support vector classifier and the support vector machine? How are these chosen above?
- Q15. How would you evaluate the support vector machine decision boundary compared to the Bayes decision boundary?