

# Module 9: Recommended Exercises

TMA4268 Statistical Learning V2022

Emma Skarstein, Daesoo Lee, Stefanie Muff  
Department of Mathematical Sciences, NTNU

March 14, 2022

---

## Problem 1

Work through the lab in Section 9.6.1 of the course book.

## Problem 2 (Book Ex.2)

We have seen that in  $p = 2$  dimensions, a linear decision boundary takes the form  $\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$ . We now investigate a non-linear decision boundary.

a) Sketch the curve

$$(1 + X_1)^2 + (2 - X_2)^2 = 4.$$

b) On your sketch, indicate the set of points for which

$$(1 + X_1)^2 + (2 - X_2)^2 > 4,$$

as well as the set of points for which

$$(1 + X_1)^2 + (2 - X_2)^2 \leq 4.$$

c) Suppose that a classifier assigns an observation to the blue class if

$$(1 + X_1)^2 + (2 - X_2)^2 > 4,$$

and to the red class otherwise. To what class is the observation  $(0, 0)$  classified?  $(-1, 1)$ ?  $(2, 2)$ ?  $(3, 8)$ ?

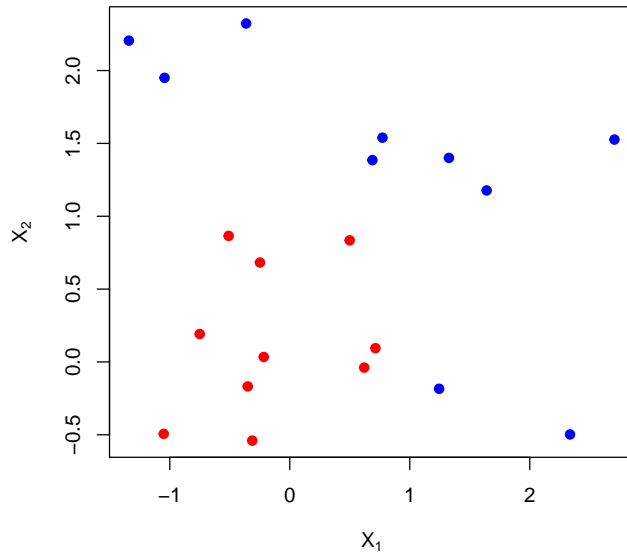
d) Argue that while the decision boundary in (c) is not linear in terms of  $X_1$  and  $X_2$ , it is linear in terms of  $X_1, X_1^2, X_2$ , and  $X_2^2$ .

## Problem 3

This problem involves plotting of decision boundaries for different kernels and it's taken from [Lab video](#).

*# code taken from video by Trevor Hastie*

```
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, xlab = expression(X[1]), ylab = expression(X[2]))
```



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

- (a) Plot the decision boundary of the `svmf` model by using the function `make.grid`. Hint: Use the predict function for the grid points and then plot the predicted values  $\{-1,1\}$  with different colors.

**R-hints:**

```
library(e1071)
svmf = svm(y ~ ..., ..., kernel = "...", cost = ..., scale = ...)
```

The following function may help you to generate a grid for plotting:

```
make.grid = function(x, n = 75) {
  # takes as input the data matrix x and number of grid points n in each direction
  # the default value will generate a 75x75 grid
  grange = apply(x, 2, range) # range for x1 and x2
  x1 = seq(from = grange[1, 1], to = grange[2, 1], length.out = n) # sequence from the lowest to the
  x2 = seq(from = grange[1, 2], to = grange[2, 2], length.out = n) # sequence from the lowest to the
  expand.grid(X1 = x1, X2 = x2) #create a uniform grid according to x1 and x2 values
}
```

- (b) On the same plot add the training points and indicate the support vectors.  
 (c) The solutions to the SVM optimization problem is given by

$$\hat{\beta} = \sum_{i \in S} \hat{\alpha}_i y_i x_i ,$$

where  $S$  is the set of the support vectors. From the `svm()` function we cannot extract  $\hat{\beta}$ , but instead we have access to  $\text{coef}_i = \hat{\alpha}_i y_i$ , and  $\hat{\beta}_0$  is given as `rho`. For more details see [here](#).

Calculate the coefficients  $\hat{\beta}_0, \hat{\beta}_1$  and  $\hat{\beta}_2$ . Then add the decision boundary and the margins using the function `abline()` on the plot from (b).

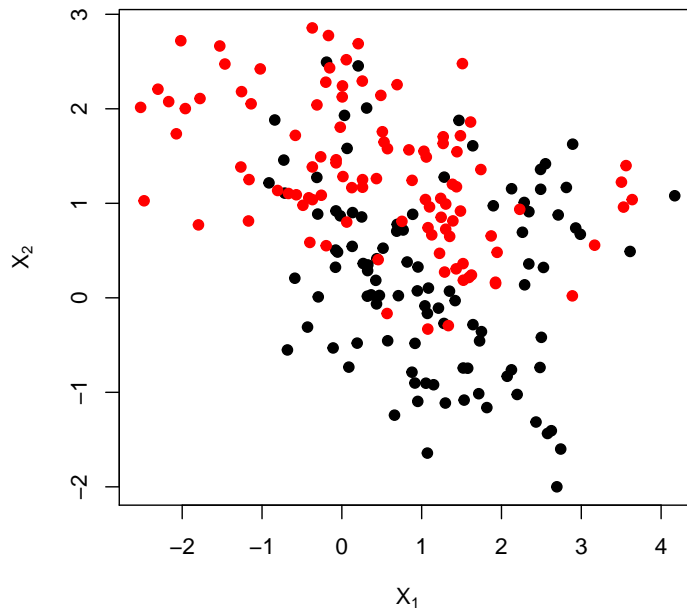
## Problem 4

Now we fit an svm model with radial kernel to the following data taken from Hastie, Tibshirani, and Friedman (2009). Use cross-validation to find the best set of tuning parameters (cost  $C$  and  $\gamma$ ). Using the same idea as in Problem 4a) plot the non-linear decision boundary, and add the training points. Furthermore if you want

to create the decision boundary curve you can use the argument `decision.values=TRUE` in the function `predict`, and then you can plot it by using the `contour()` function.

**R-hints:**

```
load(url("https://web.stanford.edu/~hastie/ElemStatLearn/datasets/ESL.mixture.rda"))
# names(ESL.mixture)
rm(x, y)
attach(ESL.mixture)
plot(x, col = y + 1, pch = 19, xlab = expression(X[1]), ylab = expression(X[2]))
```



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

To run cross-validation over a grid for  $(C, \gamma)$ , you can use a two-dimensional list of values in the `ranges` argument:

```
r.cv <- tune(svm, factor(y) ~ ., data = dat, kernel = "...", ranges = list(cost = c(...),
  gamma = c(...)))
```

For the plot:

```
xgrid = make.grid(x)
ygrid = predict(..., xgrid)
plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = 0.2)
points(x, col = y + 1, pch = 19)

# decision boundary
func = predict(..., xgrid, decision.values = TRUE)
func = attributes(func)$decision
contour(unique(xgrid[, 1]), unique(xgrid[, 2]), matrix(func, 75, 75), level = 0,
  add = TRUE) # svm boundary
```

## Problem 5 - optional (Book Ex. 7)

This problem involves the OJ data set which is part of the ISLR package.

- (a) Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.
- (b) Fit a support vector classifier to the training data using `cost=0.01`, with `Purchase` as the response and the other variables as predictors. Use the `summary()` function to produce summary statistics, and describe the results obtained.
- (c) What are the training and test error rates?
- (d) Use the `tune()` function to select an optimal cost. Consider values in the range 0.01 to 10.
- (e) Compute the training and test error rates using this new value for cost.
- (f) Repeat parts (b) through (e) using a support vector machine with a radial kernel. Use the default value for `gamma`.
- (g) Repeat parts (b) through (e) using a support vector machine with a polynomial kernel. Set `degree=2`.
- (h) Overall, which approach seems to give the best results on this data?

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning*. 2nd ed. Vol. 1. Springer series in statistics New York.