Exercises 2

29/11/2018

Example 1. Soft-margin support vector machine

Suppose that we observe *label* data $y_1, \ldots, y_n \in \{-1, 1\}$ with corresponding *covariates* $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. We wish to estimate y_i by some function $\widehat{y}(\boldsymbol{x}_i) \in \mathbb{R}^d$, for each $i = 1, \ldots, n$.

When we conduct *hyperplane dicrimination*, the function $\widehat{y}$ is taken to be of the form:

$$\widehat{y}(\boldsymbol{x}) = \text{sign}\left(\alpha + \boldsymbol{\beta}^\top \boldsymbol{x}\right) = \text{sign}\left(\boldsymbol{\theta}^\top \bar{\boldsymbol{x}}\right),$$

where $\bar{\boldsymbol{x}}_i^\top = (1, \boldsymbol{x}_i^\top)$ and $\text{sign}(a) = a/|a|$ when $a \neq 0$ and $\text{sign}(0) = 0$.

The classic *support vector machine* (SVM) problem is to obtain an optimal hyperplane of form $\widehat{y}$, by solving the optimization problem:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^{d+1}} \left\{ f(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \left[1 - y_i \left(\alpha + \boldsymbol{\beta}^\top \boldsymbol{x}_i\right)\right]_+ + \lambda \sum_{j=1}^{d} \beta_j^2 \right\},$$

where $\lambda \geq 0$ is a regularization constant.

Here, we call

$$l\left(y, \alpha + \boldsymbol{\beta}^\top \boldsymbol{x}\right) = \left[1 - y\left(\alpha + \boldsymbol{\beta}^\top \boldsymbol{x}\right)\right]_+$$

the loss between the observed *label* y and the *estimate* $\widehat{y}$.

In this example, we will construct an *optimal discriminant hyperplane* by solving the SVM problem for data arising from the following process.

For $n = 200$, let $y_1, \ldots y_{n/2} = -1$ and $y_{n/2+1}, \ldots y_n = 1$. For each i generate $\boldsymbol{x}_i$, if $y_i = -1$, then from a multivariate normal distribution with mean $\boldsymbol{\mu}_1^\top = (-1, -1)$ and covariance $\mathbf{I}$, if $y_i = 1$, then from a multivariate normal distribution with mean $\boldsymbol{\mu}_2^\top = (1, 1)$ and covariance $\mathbf{I}$.

```
# Set a random seed
set.seed(200)
# Set the value of n
nn <- 200
# Generate y labels
yy <- c(rep(-1,nn/2),rep(1,nn/2))
# Generate x covariates
XX <- rbind(matrix(rnorm(nn,-1,1),nn/2,2),
            matrix(rnorm(nn,1,1),nn/2,2))
```

Let $\bar{\boldsymbol{y}}_i = y_i \bar{\boldsymbol{x}}_i$ and let $\mathbf{Y}$ be a matrix with ith row $\bar{\boldsymbol{y}}_i$.

```
# Compute the vectors x_bar
XX_bar <- cbind(1,XX)
# Compute the matrix YY
YY <- matrix(rep(yy,3),nn,3)*XX_bar
```

We now proceed to run the MM algorithm for estimation of the optimal discriminant hyperplane. That is, at the rth iteration of our algorithm, we will compute the rth iterate $\boldsymbol{\theta}^{(r)}$ using the update rule:

$$\boldsymbol{\theta}^{(r)} = \left(\mathbf{Y}^\top \mathbf{W}_\epsilon^{(r-1)} \mathbf{Y} + 4n\lambda \bar{\mathbf{I}}_d\right)^{-1} \mathbf{Y}^\top \left(\mathbf{1} + \mathbf{W}_\epsilon^{(r-1)} \mathbf{1}\right),$$

where $\epsilon > 0$ is a small constant and $\mathbf{W}_{\epsilon}^{(r-1)}$ is a diagonal matrix with ith element

$$w_{i,\epsilon}^{(r-1)} = 1/\sqrt{\left(1 - y_i \boldsymbol{\theta}^{(r-1)\top} \bar{\boldsymbol{x}}_i\right)^2 + \epsilon},$$

$$\bar{\mathbf{I}}_d = \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{I}_d \end{bmatrix},$$

and $\mathbf{1}$ is a column vector of ones, of appropriate dimensionality.

We shall impliment the algorithm using $\epsilon = 0.01$, $lambda = 1$, and $R = 100$ iterations. We will initialize the algorithm at the value $\boldsymbol{\theta}^{\top} = \mathbf{0}$.

```
# Set the number of iterations
RR <- 100
# Set the degree of approximation epsilon
epsilon <- 0.01
# Set the level of regularization
lambda <- 1
# Make the matrix I_bar
II_bar <- diag(c(0,1,1))
# Set a starting value for theta
theta_svm <- c(0,0,0)
for (rr in 1:RR) {
  # Generate the weight matrix
  WW <- diag(c(1/sqrt((1-YY%*%matrix(theta_svm,3,1))^2+epsilon)))
  # Run an iteration of the algorithm
  theta_svm <- solve(t(YY)%*%WW%*%YY+4*nn*lambda*II_bar)%*%t(YY)%*%(matrix(1,nn,1)+WW%*%matrix(1,nn,1))
}
```

Let $\widehat{\boldsymbol{\theta}}$ be our obtained estimate for $\boldsymbol{\theta}$. We can print the value of the obtained estimate by calling the variable `theta_svm`.

```
theta_svm
```

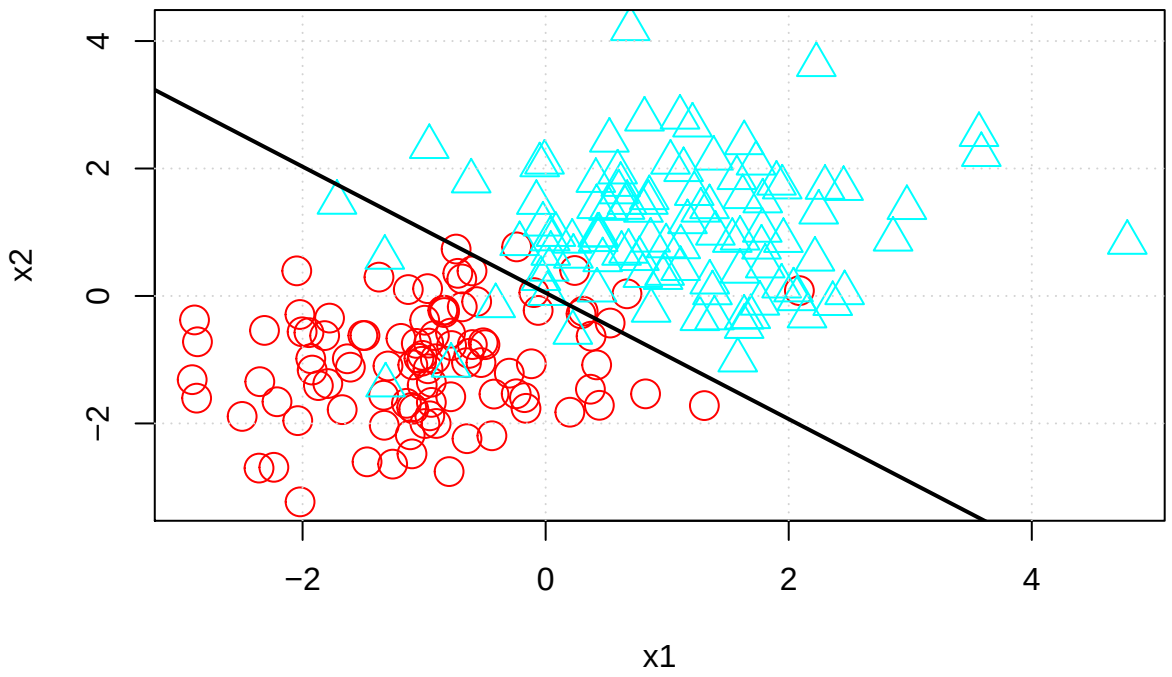
```
##             [,1]
## [1,] -0.01511106
## [2,]  0.30789056
## [3,]  0.31093530
```

Do changes in the value of λ effect the value of $\widehat{\boldsymbol{\theta}}$? What about changes in the value of ϵ?

We can plot the hyperplane $\widehat{\alpha} + \widehat{\boldsymbol{\beta}}^{\top} \boldsymbol{x} = 0$ along with our data in order to visualize the performance of the SVM discriminant hyperplane.

```
# Plot the vector x for each of the observations
plot(XX,
     col=rainbow(2)[(yy+1)/2+1],
     pch=((yy+1)/2+1),cex=2,
     xlab='x1',ylab='x2')
# Add a grid to the plot
grid()
# Create a dummy variable for the plot device
dum <- seq(-4,4,length.out = 100)
# Create a function for the hyperplane
hyper_fun <- function(x,y) {
  theta_svm[1]+theta_svm[2]*x+theta_svm[3]*y
}
# Evaluate the hyperplane at the dummy variable
```

```
eval_hyper <- outer(dum,dum,hyper_fun)
# Draw the hyperplane
contour(dum,dum,eval_hyper,level=0,add=T,lwd=2,drawlabels=F)
```



Do changes in the value of λ effect the fitted hyperplane?

We can finally assess the performance of the SVM optimal discriminant hyperplane with respect to the *classification loss rate*

$$\frac{1}{n}\sum_{i=1}^{n}\left[y_i \times \widehat{y}\left(\boldsymbol{x}_i\right) < 0\right].$$

```
# Compute the estimates y_hat
yy_hat <- sign(XX_bar%*%theta_svm)
# Compute the classification loss rate
mean(yy_hat*yy<0)
```

```
## [1] 0.06
```

Exercise 1. High dimensional SVMs

In **Example 1** the code has been specialized for the case of $d = 2$. Generalize the code above to fit an SVM optimal seperation hyperplane to any dimension $d \in \mathbb{N}$.

Run your code on the following scenario, for $R = 100$ iterations, with $\lambda = 1$ and $\epsilon = 0.01$.

Set the random seed via `set.seed(1000)`. For $n = 1000$, Generate $y_1, \dots y_{n/2} = -1$ and $y_{n/2+1}, \dots y_n = 1$. Then, for each i, generate $\boldsymbol{x}_i$, if $y_i = -1$, then from a multivariate normal distribution with mean $\boldsymbol{\mu}_1^\top = (-1, -1, -1, -1, -1)$ and covariance $\mathbf{I}$, if $y_i = 1$, then from a multivariate normal distribution with mean $\boldsymbol{\mu}_2^\top = (1, 1, 1, 1, 1)$ and covariance $\mathbf{I}$.

Example 2. Logistic regression

Logistic regression assumes the same premise as that of *SVM*. However, instead of obtaining an optimal seperation hyperplane rule of the form:

$$\widehat{y}\left(\boldsymbol{x}\right)=\text{sign}\left(\alpha+\boldsymbol{\beta}^{\top}\boldsymbol{x}\right)=\text{sign}\left(\boldsymbol{\theta}^{\top}\bar{\boldsymbol{x}}\right),$$

via to the SVM problem, one solves the *logistic regression problem*

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^{d+1}}\left\{f\left(\boldsymbol{\theta}\right)=\frac{1}{n}\sum_{i=1}^{n}\log\left[1+\exp\left(-y_i\left[\alpha+\boldsymbol{\beta}^{\top}\boldsymbol{x}_i\right]\right)\right]+\lambda\sum_{j=1}^{d}\beta_j^2\right\},$$

instead.

Notice that this problem is only different to the SVM problem by the fact that the *loss function* now has the *logistic* form

$$l\left(y,\alpha+\boldsymbol{\beta}^{\top}\boldsymbol{x}\right)=\log\left[1+\exp\left(-y\left[\alpha+\boldsymbol{\beta}^{\top}\boldsymbol{x}\right]\right)\right].$$

Using the same random seed, we generate data in the same way as in **Example 1**, and we again let $\bar{\boldsymbol{y}}_i=y_i\bar{\boldsymbol{x}}_i$ and let $\mathbf{Y}$ be a matrix with ith row $\bar{\boldsymbol{y}}_i$.

```
# Set a random seed
set.seed(200)
# Set the value of n
nn <- 200
# Generate y labels
yy <- c(rep(-1,nn/2),rep(1,nn/2))
# Generate x covariates
XX <- rbind(matrix(rnorm(nn,-1,1),nn/2,2),
            matrix(rnorm(nn,1,1),nn/2,2))
# Compute the vectors x_bar
XX_bar <- cbind(1,XX)
# Compute the matrix YY
YY <- matrix(rep(yy,3),nn,3)*XX_bar
```

Let $\boldsymbol{\theta}^{(r)}$ be the rth iterate of the MM algorithm.

Upon letting

$$\mathbf{H}=(1/4)\sum_{i=1}^{n}\bar{\boldsymbol{y}}_i\bar{\boldsymbol{y}}_i^{\top},$$

and letting $\mathbf{p}^{(\mathbf{r-1})}\in\mathbb{R}^{\mathbf{n}}$ be a vector with ith element

$$\frac{\exp\left(-\bar{\boldsymbol{y}}_i^{\top}\boldsymbol{\theta}^{(r-1)}\right)}{1+\exp\left(-\bar{\boldsymbol{y}}_i^{\top}\boldsymbol{\theta}^{(r-1)}\right)},$$

we can write the MM algorithm update as

$$\boldsymbol{\theta}^{(r)}=\left(\mathbf{H}+2n\lambda\bar{\mathbf{I}}_d\right)^{-1}\left(\mathbf{H}\boldsymbol{\theta}^{(r-1)}+\mathbf{Y}^{\top}\mathbf{p}^{(r-1)}\right).$$

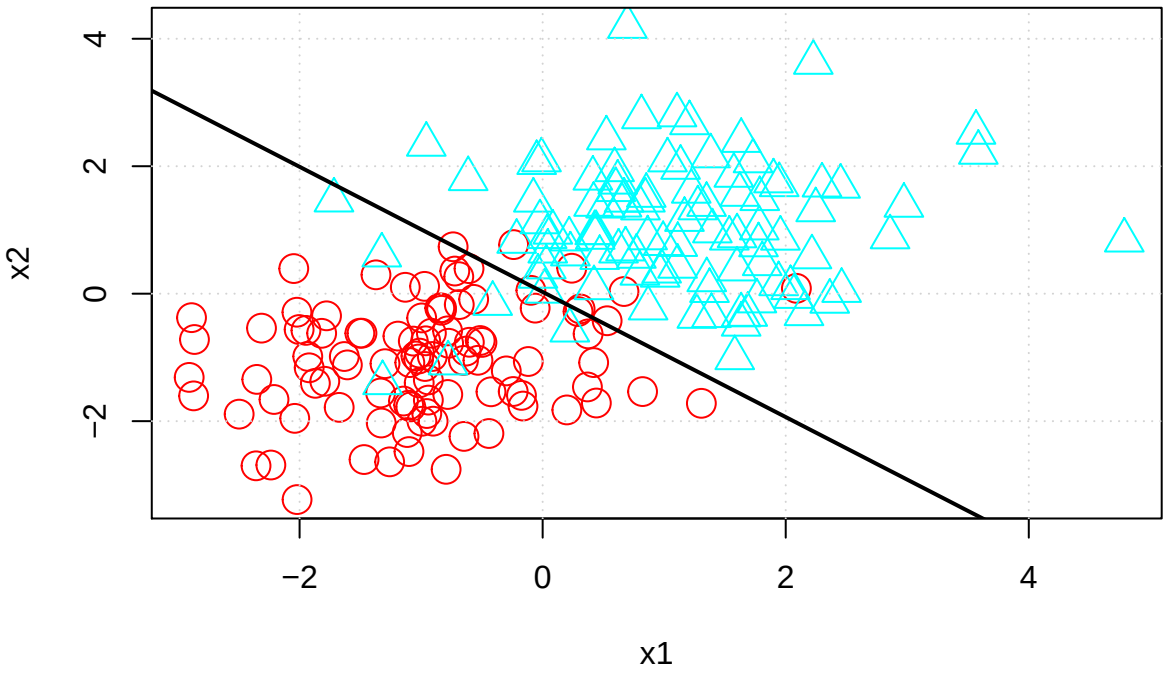
We shall impliment the algorithm using $\lambda=1$ and $R=100$ iterations. We will initialize the algorithm at the value $\boldsymbol{\theta}^{\top}=\mathbf{0}$.

```r
# Set the number of iterations
RR <- 100
# Set the level of regularization
lambda <- 1
# Make the matrix I_bar
II_bar <- diag(c(0,1,1))
# Make H matrix
HH <- (1/4)*t(YY)%*%YY
# Set a starting value for theta
theta_log <- c(0,0,0)
for (rr in 1:RR) {
  # Generate the p vector
  pp <- exp(-YY%*%theta_log)/(1+exp(-YY%*%theta_log))
  # Compute the MM algorithm
  theta_log <- solve(HH+2*nn*lambda*II_bar)%*%(HH%*%theta_log+t(YY)%*%pp)
}
```

Do changes in the value of λ effect the solution in the same way as that which was observed in Example 1?

We can plot the seperating hyperplane and compute the classification loss rate in the same way as in **Example 1**.

```r
# Plot the vector x for each of the observations
plot(XX,
     col=rainbow(2)[(yy+1)/2+1],
     pch=((yy+1)/2+1),cex=2,
     xlab='x1',ylab='x2')
# Add a grid to the plot
grid()
# Create a dummy variable for the plot device
dum <- seq(-4,4,length.out = 100)
# Create a function for the hyperplane
hyper_fun <- function(x,y) {
  theta_log[1]+theta_log[2]*x+theta_log[3]*y
}
# Evaluate the hyperplane at the dummy variable
eval_hyper <- outer(dum,dum,hyper_fun)
# Draw the hyperplane
contour(dum,dum,eval_hyper,level=0,add=T,lwd=2,drawlabels=F)
```




```
# Compute the estimates y_hat
yy_hat <- sign(XX_bar%*%theta_log)
# Compute the classification loss rate
mean(yy_hat*yy<0)
```

```
## [1] 0.06
```

Exercise 2. Newton's method

The *logistic regression problem* is more conventionally solved using *Newton's method* for optimization.

Like the MM algorithm, Newton's method is iterative. Also, like the *quadratic upper bounding* MM construction, Newton's method also makes a quadratic approximation of the function that is to be optimized, at every iteration.

That is, suppose that we wish to solve the problem:

$$\min_{\boldsymbol{\theta}} \; f\left(\boldsymbol{\theta}\right),$$

by starting at $\boldsymbol{\theta}^{(0)}$ and approximate the problem at iteration r by the problem of minimizing:

$$\bar{f}\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(r-1)}\right) \approx f\left(\boldsymbol{\theta}^{(r-1)}\right) + \frac{\partial f}{\partial \boldsymbol{\theta}}\left(\boldsymbol{\theta}^{(r-1)}\right)\left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(r-1)}\right) + \frac{1}{2}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(r-1)}\right)^{\top} \frac{\partial^2 f}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^{\top}}\left(\boldsymbol{\theta}^{(r-1)}\right)\left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(r-1)}\right),$$

which is the second-order *Taylor expansion* of $f(\boldsymbol{\theta})$.

Assuming that the *Hessian*

$$\frac{\partial^2 f}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^{\top}}\left(\boldsymbol{\theta}^{(r-1)}\right)$$

is positive definite at $\boldsymbol{\theta}^{(r-1)}$, we can find a solution to the approximate problem by solving for *first order condition*

$$\mathbf{f}^{(r-1)} + \mathbf{H}^{(r-1)}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^{(r-1)}\right) = \mathbf{0},$$

where

$$\mathbf{f}^{(r-1)} = \frac{\partial f}{\partial \boldsymbol{\theta}} \left(\boldsymbol{\theta}^{(r-1)} \right)$$

and

$$\mathbf{H}^{(r-1)} = \frac{\partial^2 f}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top} \left(\boldsymbol{\theta}^{(r-1)} \right).$$

This yields the Newton update rule

$$\boldsymbol{\theta} = \boldsymbol{\theta}^{(r-1)} - \left[\mathbf{H}^{(r-1)} \right]^{-1} \mathbf{f}^{(r-1)}.$$

In the case of logistic regression problem from **Example 2**, we can write the

$$\frac{\partial l}{\partial \boldsymbol{\theta}} (\boldsymbol{\theta}) = -\frac{\exp\left(-\bar{\boldsymbol{y}}^\top \boldsymbol{\theta}\right)}{1 + \exp\left(-\bar{\boldsymbol{y}}^\top \boldsymbol{\theta}\right)} \bar{\boldsymbol{y}},$$

and

$$\frac{\partial^2 l}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top} (\boldsymbol{\theta}) = \frac{1}{1 + \exp\left(-\bar{\boldsymbol{y}}^\top \boldsymbol{\theta}\right)} \times \frac{\exp\left(-\bar{\boldsymbol{y}}^\top \boldsymbol{\theta}\right)}{1 + \exp\left(-\bar{\boldsymbol{y}}^\top \boldsymbol{\theta}\right)} \bar{\boldsymbol{y}} \bar{\boldsymbol{y}}^\top,$$

for

$$l\left(y, \alpha + \boldsymbol{\beta}^\top \boldsymbol{x}\right) = \log\left[1 + \exp\left(-y\left[\alpha + \boldsymbol{\beta}^\top \boldsymbol{x}\right]\right)\right].$$

Using these facts, derive a Newton's algorithm for the logistic regression problem. Using fixed starting points, set seeds, and runs of increasing numbers of iterations, investigate the relative convergence properties of the two algorithms.

Example 3. Quadratic discrimination by SVM

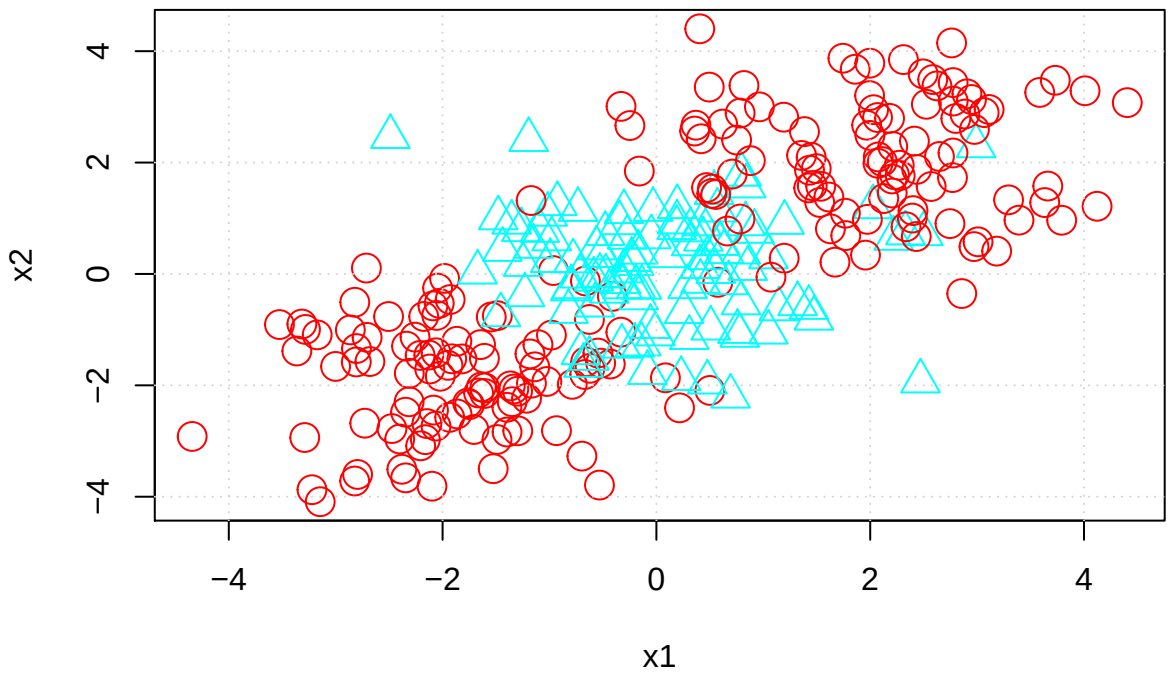
In the examples so far, the data that have been used as examples have all been clearly seperable via a linear hyperplane. However, data are often complex and linear seperation is often not possible. We consider for example the following situation.

Let $n = 300$ and let $y_1, \ldots y_{n/3} = -1$, $y_{n/3+1}, \ldots y_{2n/3} = 1$, and $y_{2n/3+1}, \ldots y_n = -1$. For $i = 1, ..., n/3$, generate $\boldsymbol{x}_i$ from a multivariate normal distribution with mean $\boldsymbol{\mu}_1^\top = (-2, -2)$ and covariance $\mathbf{I}$. For $i = n/3 + 1, ..., 2n/3$, generate $\boldsymbol{x}_i$ from a multivariate normal distribution with mean $\boldsymbol{\mu}_2^\top = (0, 0)$ and covariance $\mathbf{I}$. And for $i = n/3 + 1, ..., n$, generate $\boldsymbol{x}_i$ from a multivariate normal distribution with mean $\boldsymbol{\mu}_3^\top = (2, 2)$ and covariance $\mathbf{I}$.

The difficulty in this problem becomes apparent once the data has been plotted.

```
# Set a random seed
set.seed(300)
# Set the value of n
nn <- 300
# Generate y labels
yy <- c(rep(-1,nn/3),rep(1,nn/3),rep(-1,nn/3))
# Generate x covariates
XX <- rbind(matrix(rnorm(nn,-2,1),nn/3,2),
            matrix(rnorm(nn,0,1),nn/3,2),
            matrix(rnorm(nn,2,1),nn/3,2))
# Plot the vector x for each of the observations
plot(XX,
     col=rainbow(2)[(yy+1)/2+1],
     pch=((yy+1)/2+1),cex=2,
```

```
     xlab='x1',ylab='x2')
# Add a grid to the plot
grid()
```



We can attempt to fit the SVM optimal seperation hyperplane from **Example 1**.

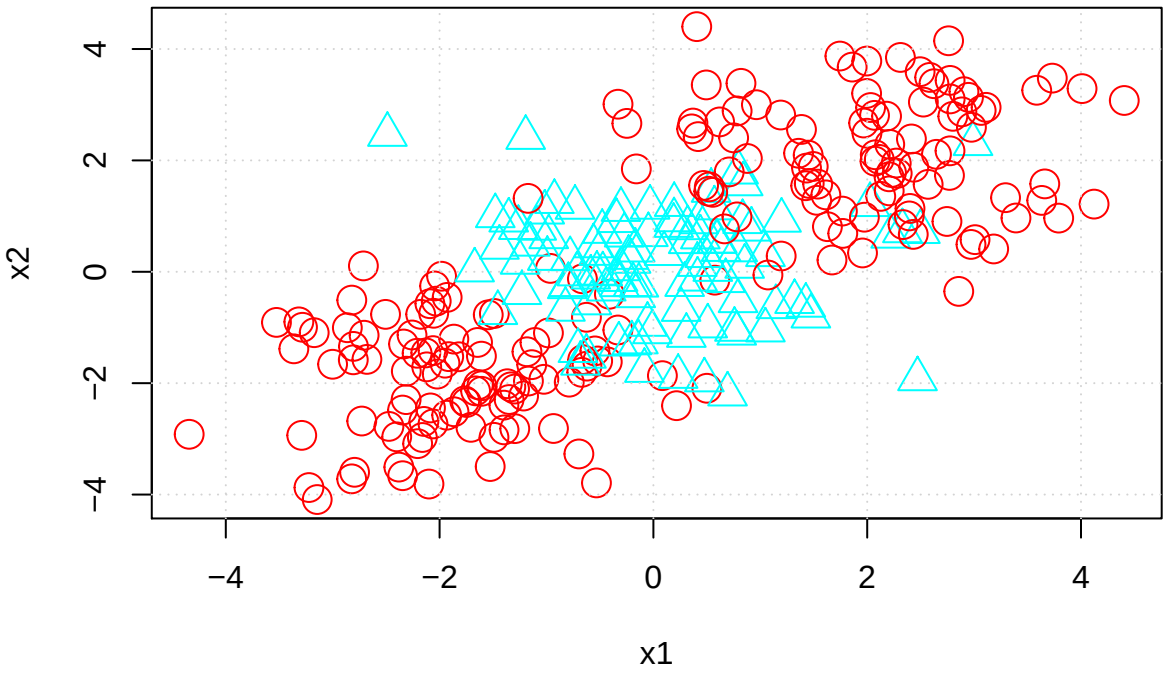
What do you expect to be the outcome of fit?

```
# Compute the vectors x_bar
XX_bar <- cbind(1,XX)
# Compute the matrix YY
YY <- matrix(rep(yy,3),nn,3)*XX_bar
# Set the number of iterations
RR <- 100
# Set the degree of approximation epsilon
epsilon <- 0.01
# Set the level of regularization
lambda <- 1
# Make the matrix I_bar
II_bar <- diag(c(0,1,1))
# Set a starting value for theta
theta_svm <- c(0,0,0)
for (rr in 1:RR) {
  # Generate the weight matrix
  WW <- diag(c(1/sqrt((1-YY%*%matrix(theta_svm,3,1))^2+epsilon)))
  # Run an iteration of the algorithm
  theta_svm <- solve(t(YY)%*%WW%*%YY+4*nn*lambda*II_bar)%*%t(YY)%*%(matrix(1,nn,1)+WW%*%matrix(1,nn,1))
}
```

A plot of the hyperplane is given as follows.

```
# Plot the vector x for each of the observations
plot(XX,
     col=rainbow(2)[(yy+1)/2+1],
```

```
     pch=((yy+1)/2+1),cex=2,
     xlab='x1',ylab='x2')
# Add a grid to the plot
grid()
# Create a dummy variable for the plot device
dum <- seq(-4,4,length.out = 100)
# Create a function for the hyperplane
hyper_fun <- function(x,y) {
  theta_svm[1]+theta_svm[2]*x+theta_svm[3]*y
}
# Evaluate the hyperplane at the dummy variable
eval_hyper <- outer(dum,dum,hyper_fun)
# Draw the hyperplane
contour(dum,dum,eval_hyper,add=T,lwd=2,drawlabels=F,levels = 0)
```



Unfortunately, this plot is not very revealing as the seperating hyperplane cannot be seen anywhere. However, a more revealing result regarding the hyperplane appears when we compute the classification loss rate.

```
# Compute the estimates y_hat
yy_hat <- sign(XX_bar%*%theta_svm)
# Compute the classification loss rate
mean(yy_hat*yy<0)
```

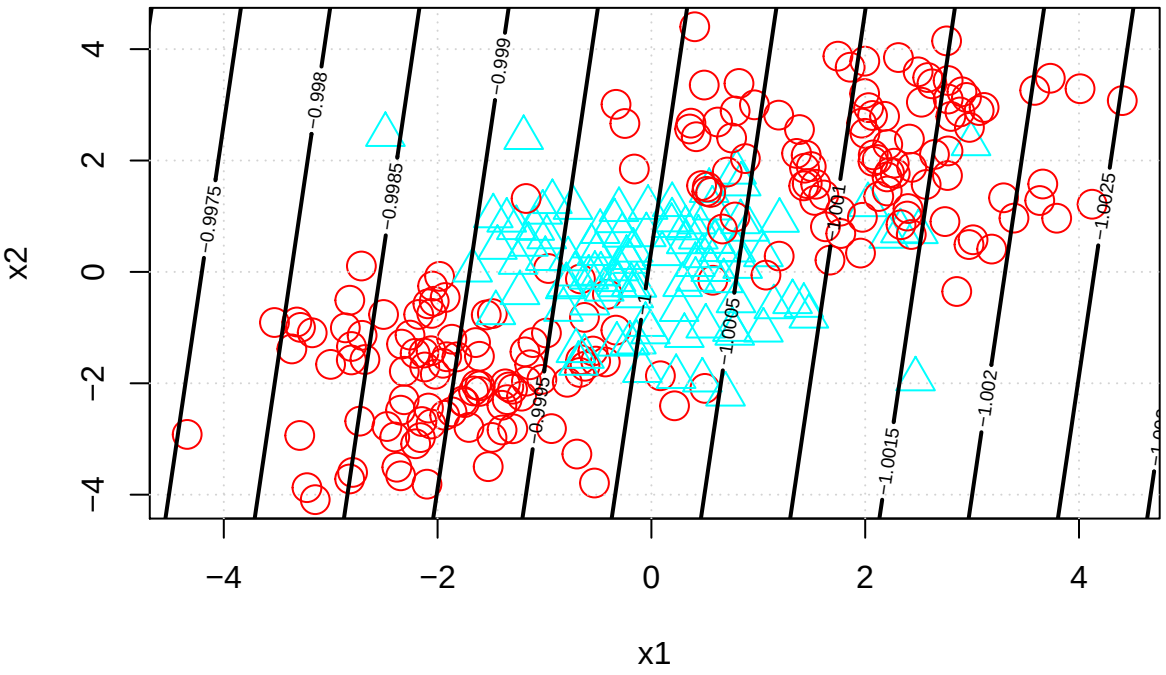
```
## [1] 0.3333333
```

The value suggests that the hyperplane is set so that all of the labels y_i are predicted to be $\widehat{y}(\boldsymbol{x}_i) = -1$. We can see this by plotting the contours of the function

$$h(\boldsymbol{x}) = \widehat{\alpha} + \widehat{\boldsymbol{\beta}}^{\top}\boldsymbol{x}.$$

```
# Plot the vector x for each of the observations
plot(XX,
     col=rainbow(2)[(yy+1)/2+1],
     pch=((yy+1)/2+1),cex=2,
```

```
     xlab='x1',ylab='x2')
# Add a grid to the plot
grid()
# Create a dummy variable for the plot device
dum <- seq(-5,5,length.out = 100)
# Create a function for the hyperplane
hyper_fun <- function(x,y) {
  theta_svm[1]+theta_svm[2]*x+theta_svm[3]*y
}
# Evaluate the hyperplane at the dummy variable
eval_hyper <- outer(dum,dum,hyper_fun)
# Draw the hyperplane
contour(dum,dum,eval_hyper,add=T,lwd=2,drawlabels=T)
```



Notice the negativity of each of the contours. Was this what you expected?

We will now demonstrate how the linear discriminant method of *optimal hyperplane discrimination* can be turned into a *nonlinear method*. Namely, we will demonstrate a how we can conduct **quadratic** classification.

To do this, for each $i \in 1, \ldots, n$, we simply compliment our covariate vector $\boldsymbol{x}_i^\top = (x_{i1}, x_{i2})$ by expanding it with three additional covariate elements $x_{i3} = x_{i1}^2$, $x_{i4} = x_{i2}^2$, and $x_{i5} = x_{i1}x_{i2}$. We now have a $d = 5$ dimensional covariate vector

$$\boldsymbol{x}_i^\top = (x_{i1}, x_{i2}, x_{i3}, x_{i4}).$$

```
# Generate the two new covariates
XX <- cbind(XX,XX[,1]^2,XX[,2]^2,XX[,1]*XX[,2])
```

We now solve the SVM problem for $d = 5$ instead of $d = 2$, for our newly constructed *extended* data set, as per **Example 1**.

```
# Compute the vectors x_bar
XX_bar <- cbind(1,XX)
# Compute the matrix YY
YY <- matrix(rep(yy,6),nn,6)*XX_bar
```

```
# Set the number of iterations
RR <- 100
# Set the degree of approximation epsilon
epsilon <- 0.01
# Set the level of regularization
lambda <- 1
# Make the matrix I_bar
II_bar <- diag(c(0,1,1,1,1,1))
# Set a starting value for theta
theta_svm <- c(0,0,0,0,0,0)
for (rr in 1:RR) {
  # Generate the weight matrix
  WW <- diag(c(1/sqrt((1-YY%*%matrix(theta_svm,6,1))^2+epsilon)))
  # Run an iteration of the algorithm
  theta_svm <- solve(t(YY)%*%WW%*%YY+4*nn*lambda*II_bar)%*%t(YY)%*%(matrix(1,nn,1)+WW%*%matrix(1,nn,1))
}
```

Let

$$\widehat{\boldsymbol{\theta}}^{\top} = \left(\widehat{\alpha}, \widehat{\boldsymbol{\beta}}^{\top}\right) = \left(\widehat{\alpha}, \widehat{\beta}_1, \widehat{\beta}_2, \widehat{\beta}_3, \widehat{\beta}_4, \widehat{\beta}_5\right),$$

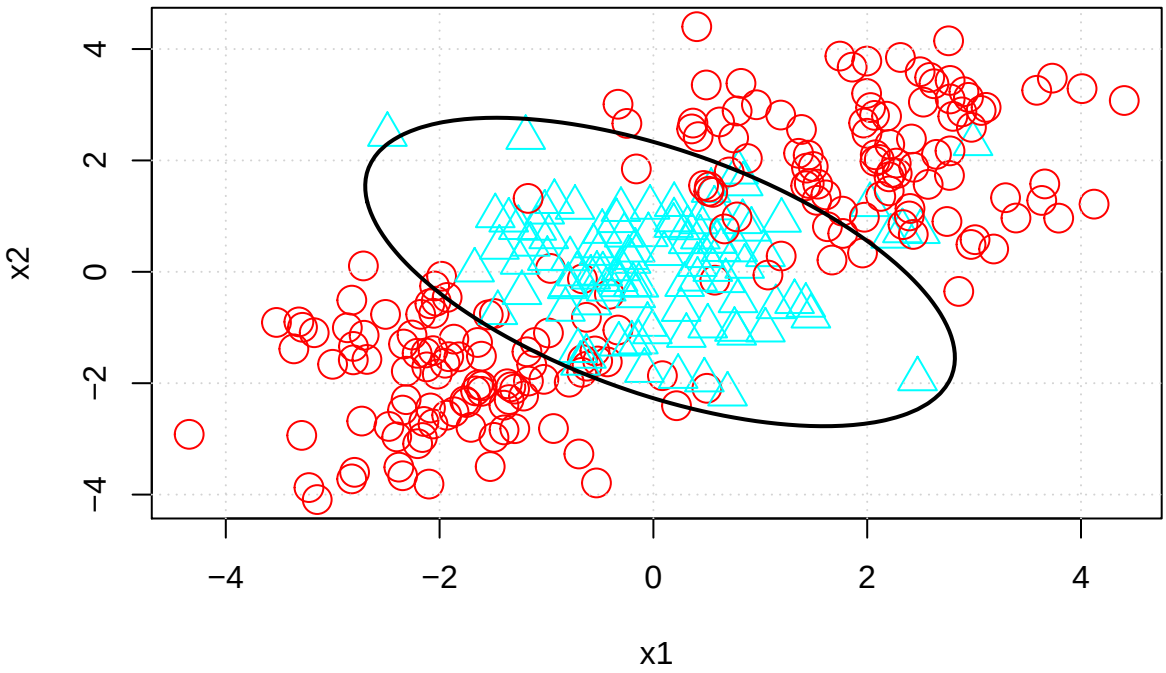
be our obtained optimal seperating hyperplane parameter vector. We have now obtained a seperating *manifold* of the form

$$h\left(x_1, x_2\right) = \widehat{\alpha} + \widehat{\beta}_1 x_1 + \widehat{\beta}_2 x_2 + \widehat{\beta}_3 x_1^2 + \widehat{\beta}_4 x_2^2 + \widehat{\beta}_5 x_1 x_2.$$

We shall plot the manifold in order to assess the performance of the usual rule:

$$\widehat{y}\left(\boldsymbol{x}\right) = \text{sign}\left(\boldsymbol{\theta}^{\top} \bar{\boldsymbol{x}}\right).$$

```
# Plot the vector x for each of the observations
plot(XX,
     col=rainbow(2)[(yy+1)/2+1],
     pch=((yy+1)/2+1),cex=2,
     xlab='x1',ylab='x2')
# Add a grid to the plot
grid()
# Create a dummy variable for the plot device
dum <- seq(-5,5,length.out = 100)
# Create a function for the hyperplane
hyper_fun <- function(x,y) {
  theta_svm[1]+theta_svm[2]*x+theta_svm[3]*y+theta_svm[4]*x^2+theta_svm[5]*y^2+theta_svm[6]*x*y
}
# Evaluate the hyperplane at the dummy variable
eval_hyper <- outer(dum,dum,hyper_fun)
# Draw the hyperplane
contour(dum,dum,eval_hyper,add=T,lwd=2,drawlabels=F,levels = 0)
```



We once again calculate the average classification loss in order to assess whether there has been an improvement to the linear discrimant result.

```
# Compute the estimates y_hat
yy_hat <- sign(XX_bar%*%theta_svm)
# Compute the classification loss rate
mean(yy_hat*yy<0)
```

```
## [1] 0.14
```

Exercise 3. Warm starts

When training machine learning algorithms, such as *optimal seperating hyerplanes*, using iterative algorithms such as the MM algorithm, it is often desirable to initialize the algorithm at a solution that is close to the eventual optimal value.

This can be done by solving a simpler problem in order to provide a *warm start* for the more complicated problem. As an example, in all of our previous exercises, we have used $\boldsymbol{\theta}^{(0)} = \mathbf{0}$ as our initializations. However, when the solution is far from the origin, this will prolong the convergence of the algorithm.

From the lecture, we know that the *least squares* optimal seperating hyperplane problem, defined as

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^{d+1}} \left\{ f(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \left[1 - y_i \left(\alpha + \boldsymbol{\beta}^\top \boldsymbol{x}_i \right) \right]^2 + \lambda \sum_{j=1}^{d} \beta_j^2 \right\},$$

can be solved in closed form.

The solution is given as

$$\boldsymbol{\theta}^* = \left(\mathbf{Y}^\top \mathbf{Y} + n\lambda \bar{\mathbf{I}}_d \right)^{-1} \mathbf{Y}^\top \mathbf{1},$$

and provides a good approximation for the optimal parameter vector $\widehat{\boldsymbol{\theta}}$ of either the SVM or the logistic regression problem.

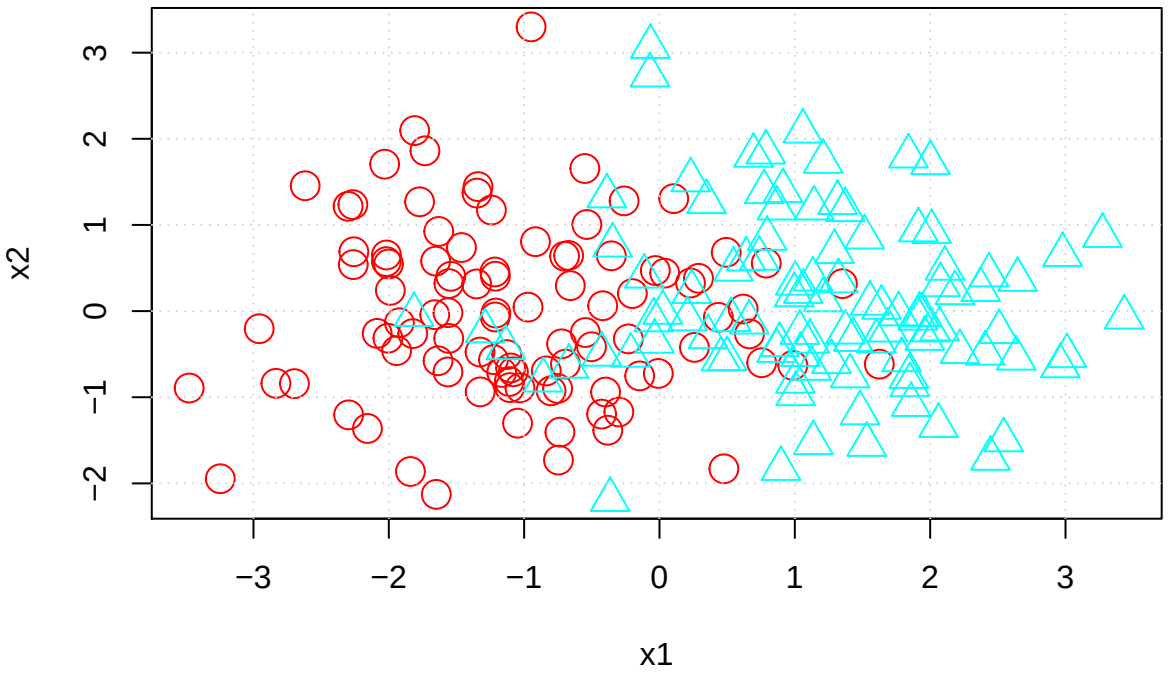
Obtain the least squares optimal seperating hyperplane for the data from **Example 1**. Using the least squares solution as a warm start, comment on whether it reduces the number of iterations required for convergence in the SVM and the logistic regression problems.

Exercise 4. Sparse SVM

Consider the following data. Set the random seed using `set.seed(3000)`. For $n = 200$, let $y_1, \ldots y_{n/2} = -1$ and $y_{n/2+1}, \ldots y_n = 1$. For each i generate $\boldsymbol{x}_i$, if $y_i = -1$, then from a multivariate normal distribution with mean $\boldsymbol{\mu}_1^\top = (-1, 0)$ and covariance $\mathbf{I}$, if $y_i = 1$, then from a multivariate normal distribution with mean $\boldsymbol{\mu}_2^\top = (1, 0)$ and covariance $\mathbf{I}$.

A plot of the data is provided below.

```
# Set a random seed
set.seed(3000)
# Set the value of n
nn <- 200
# Generate y labels
yy <- c(rep(-1,nn/2),rep(1,nn/2))
# Generate x covariates
XX <- rbind(cbind(rnorm(nn/2,-1,1),rnorm(nn/2,0,1)),
            cbind(rnorm(nn/2,1,1),rnorm(nn/2,0,1)))
# Plot the vector x for each of the observations
plot(XX,
     col=rainbow(2)[(yy+1)/2+1],
     pch=((yy+1)/2+1),cex=2,
     xlab='x1',ylab='x2')
# Add a grid to the plot
grid()
```



It is not difficult to conclude that the data permits a seperating hyperplane that depends on only one of the two covariate elements x_1 and x_2. That is, the estimated solution should be *sparse* and have either $\widehat{\beta}_1 = 0$ or $\widehat{\beta}_2 = 0$.

For some level of $\lambda > 0$ it is possible to obtain such a solution by solving the *LASSO regularized SVM problem*:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^{d+1}} \left\{ f(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \left[1 - y_i \left(\alpha + \boldsymbol{\beta}^\top \boldsymbol{x}_i \right) \right]_+ + \lambda \sum_{j=1}^{d} |\beta_j| \right\}.$$

Using the code from **Example 1** and from **Exercises 1**, deduce an MM algorithm for solving this problem.