ST4231 Assignment

A0102680R

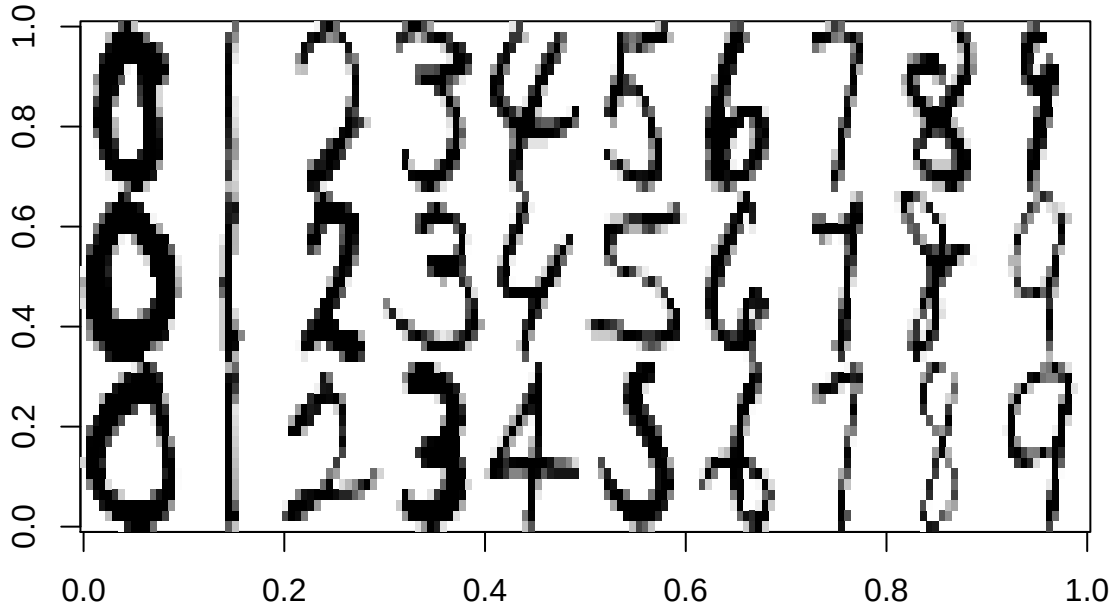
Question 1

```
library(ElemStatLearn)
data1=data(zip.train)

findRows <- function(zip, n) {
  # Find n (random) rows with zip representing 0,1,2,...,9
  res <- vector(length=10, mode="list")
  names(res) <- 0:9
  ind <- zip[,1]
  for (j in 0:9) {
    res[[j+1]] <- sample( which(ind==j), n ) }
  return(res) }

# Making a plot like that on page 4:
digits <- vector(length=10, mode="list")
names(digits) <- 0:9
rows <- findRows(zip.train, 3)
for (j in 0:9) {
  digits[[j+1]] <- do.call("cbind", lapply(as.list(rows[[j+1]]),
                                           function(x) zip2image(zip.train, x)) )
}
```

```
## [1] "digit  0  taken"
## [1] "digit  0  taken"
## [1] "digit  0  taken"
## [1] "digit  1  taken"
## [1] "digit  1  taken"
## [1] "digit  1  taken"
## [1] "digit  2  taken"
## [1] "digit  2  taken"
## [1] "digit  2  taken"
## [1] "digit  3  taken"
## [1] "digit  3  taken"
## [1] "digit  3  taken"
## [1] "digit  4  taken"
## [1] "digit  4  taken"
## [1] "digit  4  taken"
## [1] "digit  5  taken"
## [1] "digit  5  taken"
## [1] "digit  5  taken"
## [1] "digit  6  taken"
## [1] "digit  6  taken"
## [1] "digit  6  taken"
## [1] "digit  7  taken"
## [1] "digit  7  taken"
## [1] "digit  7  taken"
## [1] "digit  8  taken"
```

```
## [1] "digit  8  taken"
## [1] "digit  8  taken"
## [1] "digit  9  taken"
## [1] "digit  9  taken"
## [1] "digit  9  taken"
```

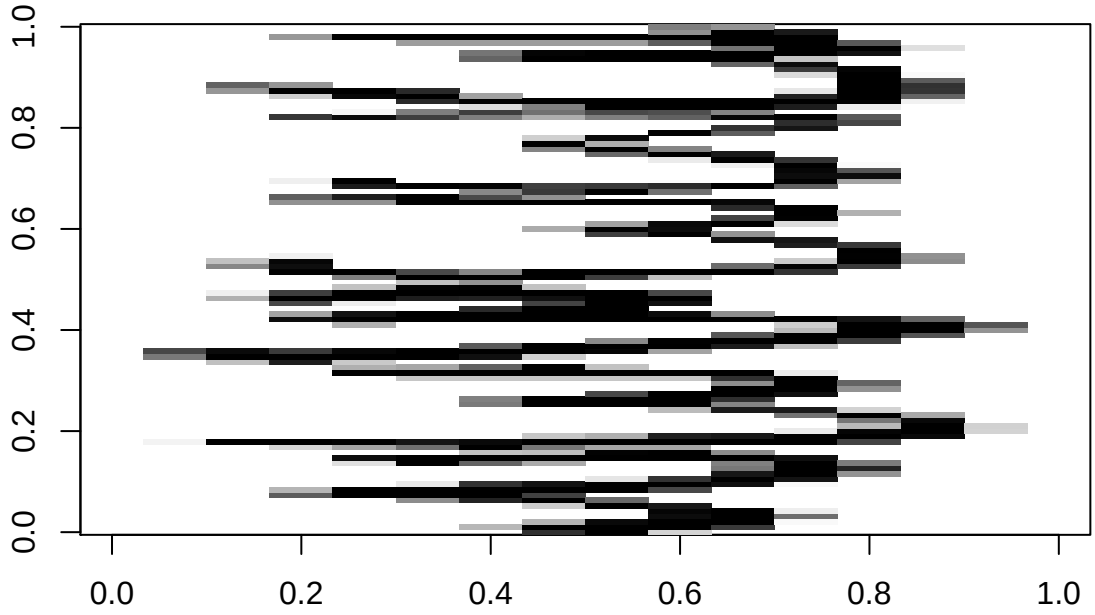
```
im <- do.call("rbind", digits)
image(im, col=gray(256:0/256), zlim=c(0,1), xlab="", ylab="" )
```




```
findRows <- function(zip, n) {
  # Find n (random) rows with zip representing 0,1,2,...,9
  res <- vector(length=10, mode="list")
  names(res) <- 0:9
  ind <- zip[,1]
  for (j in 3:3) {
    res[[j+1]] <- sample( which(ind==j), n ) }
  return(res) }

# Making a plot like that on page 4:
digits <- vector(length=10, mode="list")
names(digits) <- 0:9
rows <- findRows(zip.train, 6)
for (j in 3:3) {
  digits[[j+1]] <- do.call("cbind", lapply(as.list(rows[[j+1]]),
                                           function(x) zip2image(zip.train, x)) )
}
```

```
## [1] "digit  3  taken"
## [1] "digit  3  taken"
## [1] "digit  3  taken"
## [1] "digit  3  taken"
## [1] "digit  3  taken"
## [1] "digit  3  taken"
```

```
im <- do.call("rbind", digits)
image(im, col=gray(256:0/256), zlim=c(0,1), xlab="", ylab="" )
```




```
print(library(ElemStatLearn)
train.data <- data(zip.train)
test.data <- data(zip.test)
z_matrix=matrix(as.numeric(train.data[100,-1]),28,28)
image(z_matrix)

digits <- seq(0,9)
digits.pool=list(0)

for (i in 0:9) {

  index <- which(train.data$label ==i)
  digits.pool[[i+1]]=train.data[index,]

}
Means.digits <- sapply(digits.pool, function(x) colMeans(as.matrix(x[-1,])))
mean_cen_digits <- list(0)
```

```
for(j in 1:10){
  mu <- matrix(1,4132,1) %*% t(Means.digits[-1,j])

  mean_cen_digits[[j]] <- digits.pool[[j]][,-1]-mu

}
FuncSVD <- function(val){
  out_mat <- as.matrix(val)
  output <- svd(out_mat)
  return(output)

}
digits.svd <- lapply(mean_cen_digits, FuncSVD)

umatrix=matrix(as.numeric(digits.svd[[4]]$v[,1]),28,28)
image(umatrix)

normal.test <- list(0)
for(i in 1:10){
  normal.test[[i]] <- test.data- (Means.digits[,i] %*% matrix(1,1,10))
}
new_digits=list(0)
PC_train=list(0)
for(k in 1:10){
  new_digits[[k]] <-  as.matrix((normal.test[[k]])) %*% as.matrix(digits.svd[[k]]$v[,1:30])
  PC_train[[k]] <- t(as.matrix(digits.svd[[k]]$v[,1:30])) %*% t(as.matrix(mean_cen_digits[[k]]))

})
```

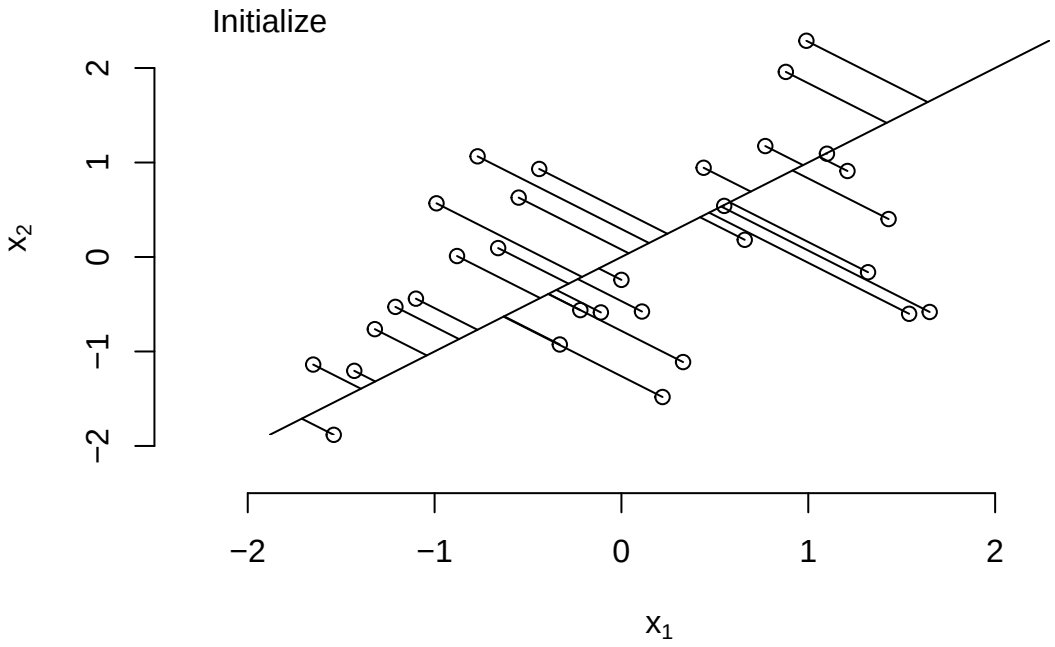
Question 2

```
## input bivariate data
data <-read.table("C:/Users/Ethan/Desktop/Assignment/principalcurve-data.txt")
x1=data$x1
x2 =data$x2
x <-scale(cbind(x1,x2))
x
```

```
##               x1          x2
##  [1,] -1.6497800 -1.13855848
##  [2,] -1.5397947 -1.88216305
##  [3,] -1.4298094 -1.20492031
##  [4,] -1.3198240 -0.76378657
##  [5,] -1.2098387 -0.52805938
##  [6,] -1.0998534 -0.44109308
##  [7,] -0.9898680  0.56812003
##  [8,] -0.8798827  0.01024691
##  [9,] -0.7698974  1.06536067
## [10,] -0.6599120  0.09427166
## [11,] -0.5499267  0.62847029
## [12,] -0.4399413  0.93157922
## [13,] -0.3299560 -0.92593221
```

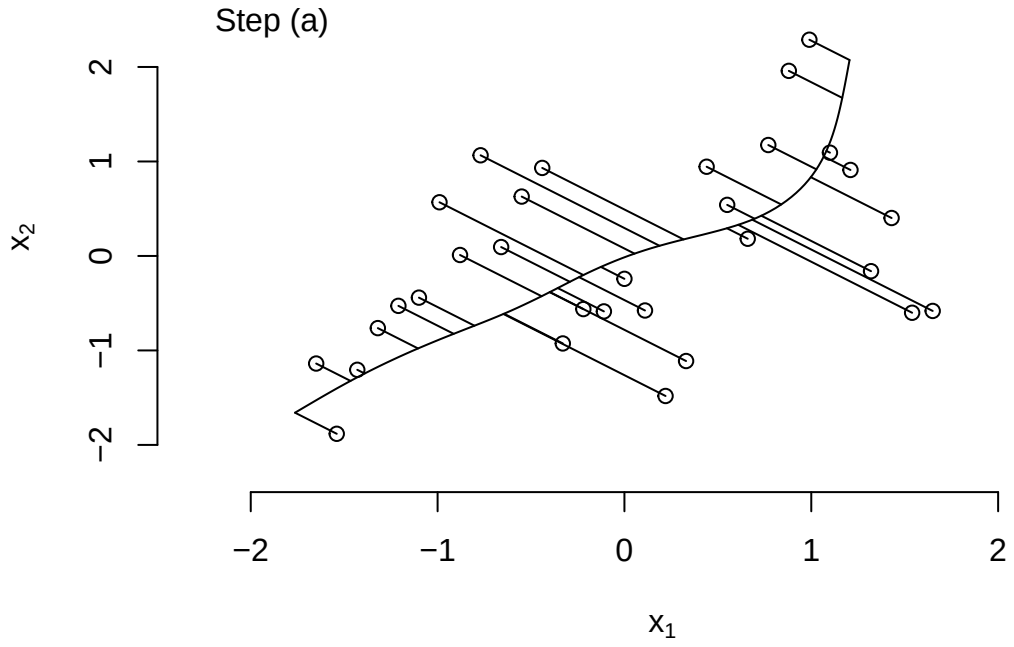
```
## [14,] -0.2199707 -0.56209194
## [15,] -0.1099853 -0.58683213
## [16,]  0.0000000 -0.24220849
## [17,]  0.1099853 -0.57719165
## [18,]  0.2199707 -1.48188543
## [19,]  0.3299560 -1.11209431
## [20,]  0.4399413  0.94431814
## [21,]  0.5499267  0.54062374
## [22,]  0.6599120  0.18188720
## [23,]  0.7698974  1.17403386
## [24,]  0.8798827  1.95898817
## [25,]  0.9898680  2.28849962
## [26,]  1.0998534  1.09309109
## [27,]  1.2098387  0.91013091
## [28,]  1.3198240 -0.16127927
## [29,]  1.4298094  0.40153770
## [30,]  1.5397947 -0.60120506
## [31,]  1.6497800 -0.58185788
## attr(,"scaled:center")
##            x1            x2
##  0.000000e+00 -9.677419e-10
## attr(,"scaled:scale")
## x1 x2
##  1  1
```

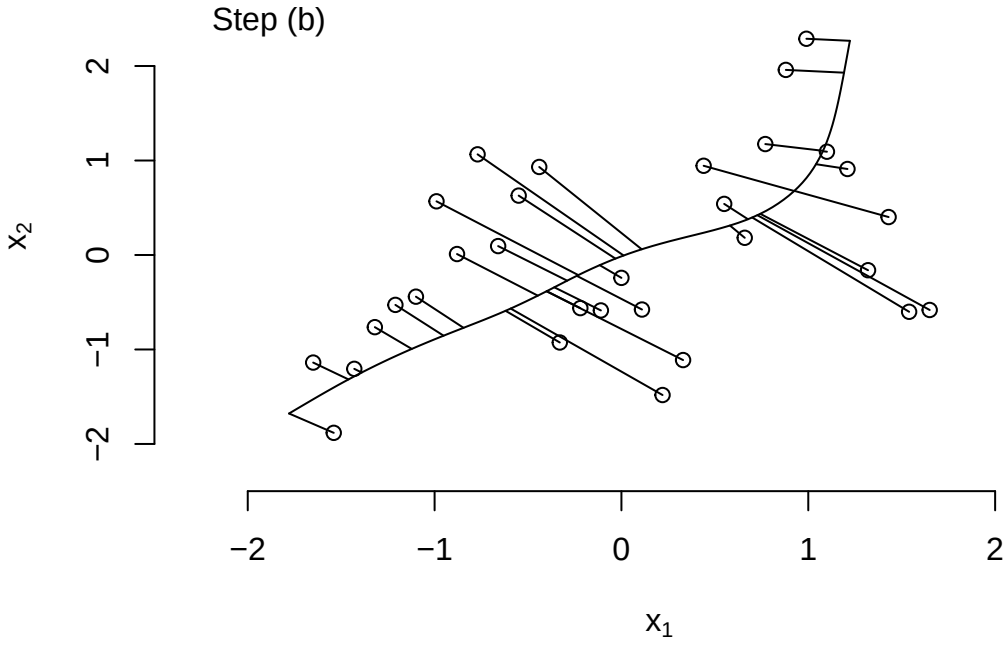
```
alim <- extendrange(x, f=0.1)
alim_ <- range(x)

## plot centered data
plot(x[,1], x[,2], bty='n',
     xlab=expression(x[1]),
     ylab=expression(x[2]),
     xlim=alim, ylim=alim)
legend("topleft", legend=c("Initialize"), bty="n")

## plot first principal component line
svdx <- svd(x)
clip(alim_[1],alim_[2],alim_[1],alim_[2])
with(svdx, abline(a=0, b=v[2,1]/v[1,1]))

## plot projections of each point onto line
z1 <- with(svdx, x%*%v[,1]%*%t(v[,1]))
segments(x0=x[,1],y0=x[,2],
         x1=z1[,1],y1=z1[,2])
```

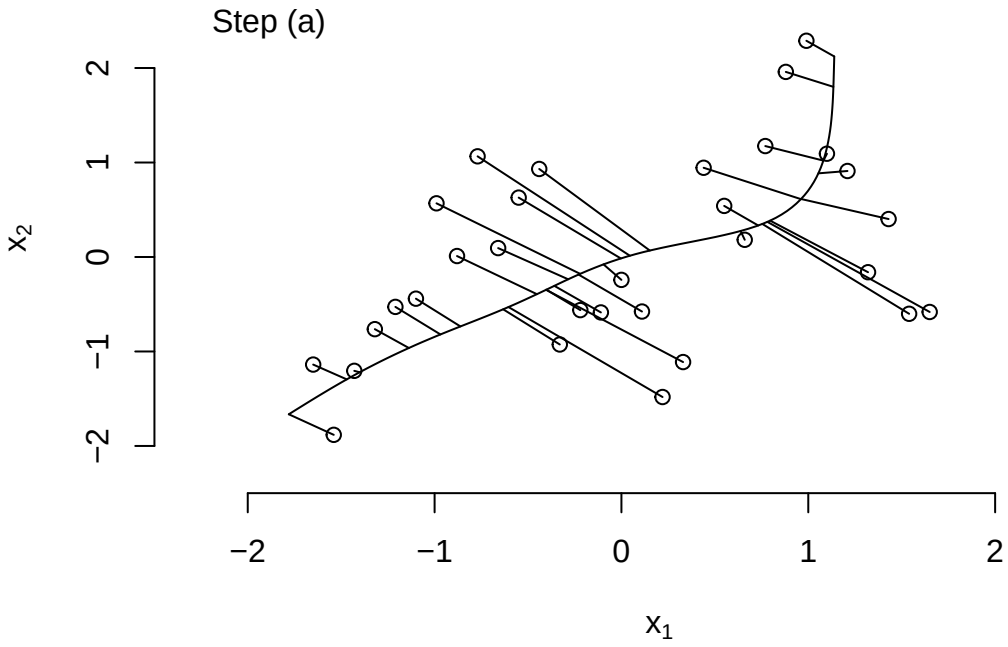


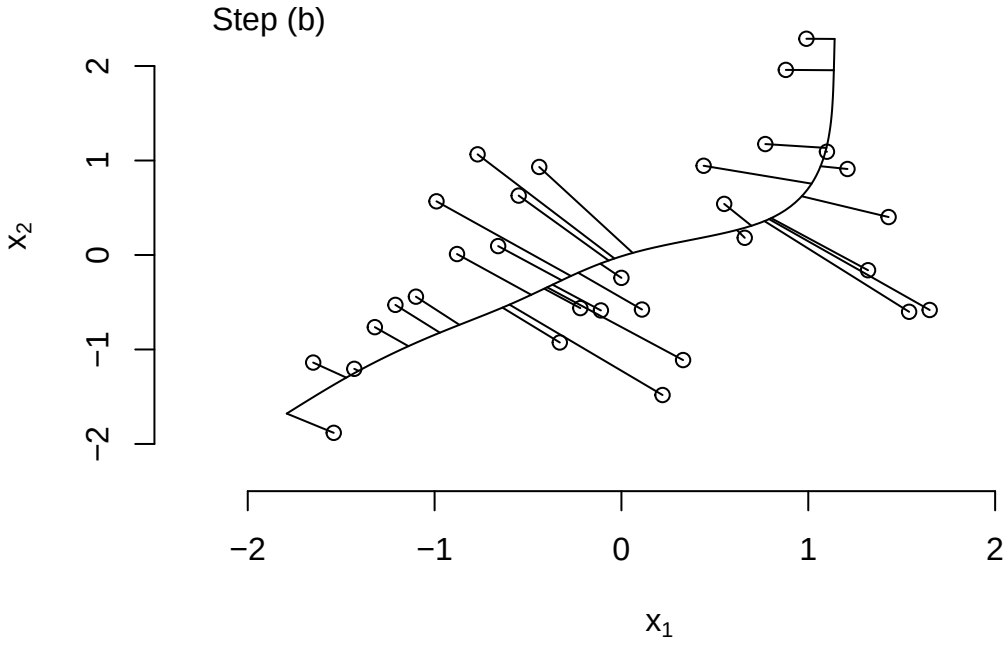

```
## compute initial lambda (arc-lengths associated with
## orthogonal projections of data onto curve)
lam <- with(svdx, as.numeric(u[,1]*d[1]))

for(itr in 1:3) {

  #### step (a) of iterative algorithm ####

  ## compute scatterplot smoother in either dimension
  ## increase 'df' to make the curve more flexible
  fit1 <- smooth.spline(x=lam, y=x[,1], df=4)
  fit2 <- smooth.spline(x=lam, y=x[,2], df=4)

  ## plot data and the principal curve for a sequence of lambdas
  plot(x[,1], x[,2], bty='n',
       xlab=expression(x[1]),
       ylab=expression(x[2]),
       xlim=alim, ylim=alim)
  legend("topleft", legend=c("Step (a)"), bty="n")
  seq_lam <- seq(min(lam),max(lam),length.out=100)
  lines(predict(fit1, seq_lam)$y, predict(fit2, seq_lam)$y)

  ## show points along curve corresponding
  ## to original lambdas
  z1 <- cbind(predict(fit1, lam)$y, predict(fit2, lam)$y)
  segments(x0=x[,1],y0=x[,2],
```

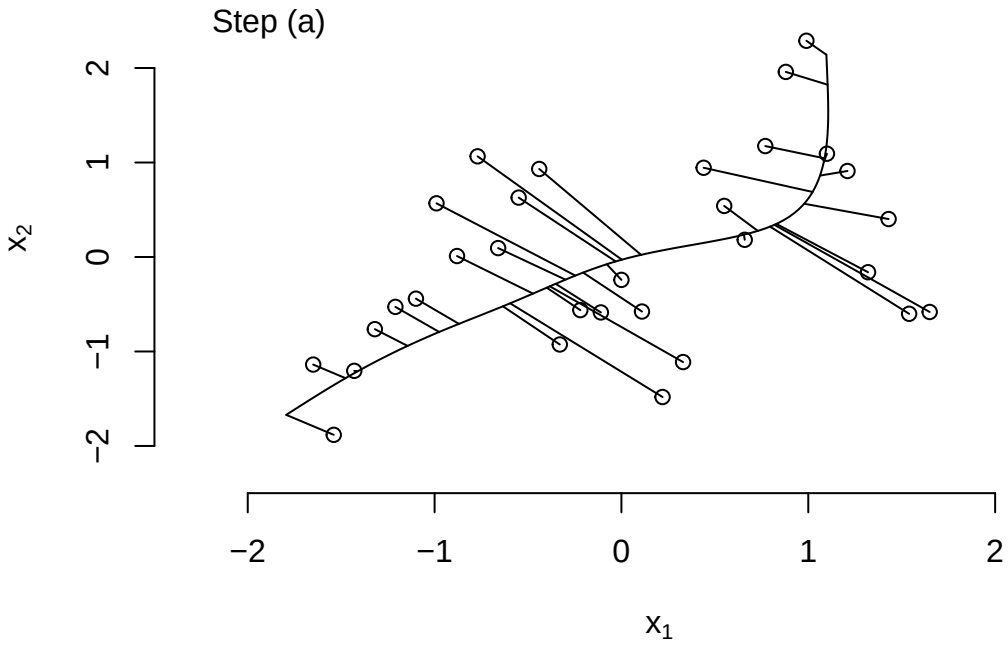
```
          x1=z1[,1],y1=z1[,2])


  #### step (b) of iterative algorithm ####

  ## recompute lambdas
  euc_dist <- function(l, x, f1, f2)
    sum((c(predict(f1, l)$y, predict(f2, l)$y) - x)^2)
  lam <- apply(x,1,function(x0) optimize(euc_dist,
    interval=extendrange(lam, f=0.50),
    x=x0, f1=fit1, f2=fit2)$minimum)


  ## show projections associated with recomputed lambdas
  plot(x[,1], x[,2], bty='n',
       xlab=expression(x[1]),
       ylab=expression(x[2]),
       xlim=alim, ylim=alim)
  legend("topleft", legend=c("Step (b)"), bty="n")
  seq_lam <- seq(min(lam),max(lam),length.out=100)
  lines(predict(fit1, seq_lam)$y, predict(fit2, seq_lam)$y)
  z1 <- cbind(predict(fit1, lam)$y, predict(fit2, lam)$y)
  segments(x0=x[,1],y0=x[,2],
           x1=z1[,1],y1=z1[,2])
}
```

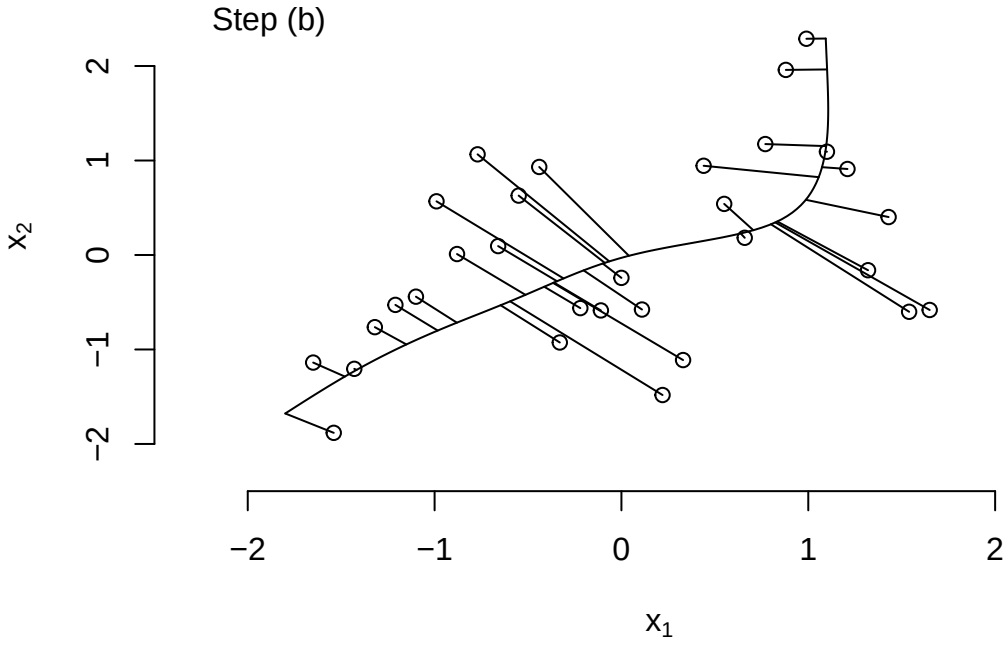



Step (b)
x2
2
1
0
−1
−2
−2
−1
0
1
2
x1


Step (a)
x2
2
1
0
−1
−2
−2
−1
0
1
2
x1


Step (b)
x2
2
1
0
−1
−2
−2
−1
0
1
2
x1


Step (a)
x2
2
1
0
−1
−2
−2
−1
0
1
2
x1


Step (b)
x2
2
1
0
−1
−2
−2
−1
0
1
2
x1