HW8

Autumn Li

11/23/2016

1. Run the following code block to create synthetic regression data, with 100 observations and 10 predictor variables:

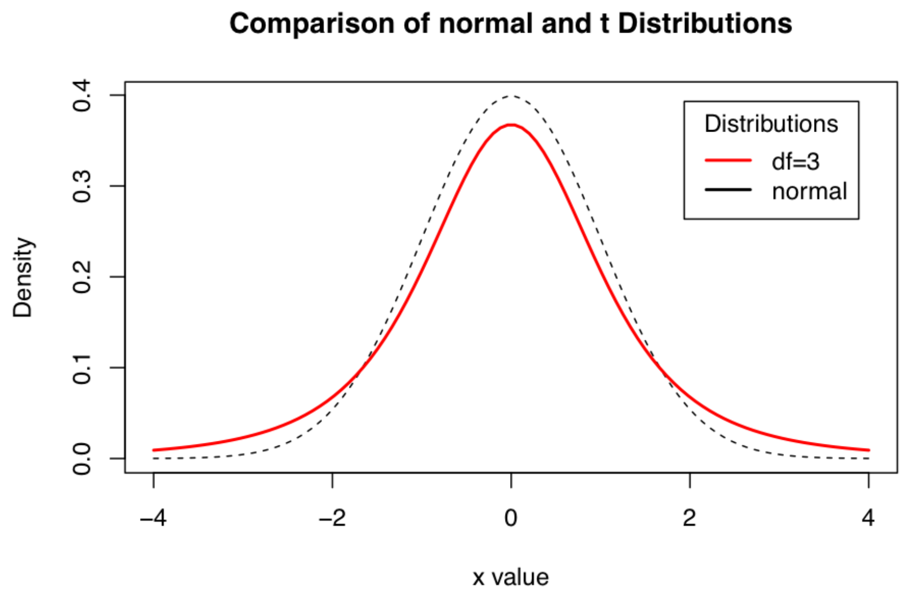
```
n <- 100
p <- 10
s <- 3
set.seed(0)
x <- matrix(rnorm(n*p), n, p)
b <- c(-0.7, 0.7, 1, rep(0, p-s))
y <- x %*% b + rt(n, df=2)
data = data.frame(x,y)
corr = cor(data,y= NULL,use="complete.obs")
# From the above correlation, I pick x1, x3 and x4.
```

2. Verify this by plotting the normal density and the t-density on the same plot, with the latter having 3 degrees of freedom.

```
x1 <- seq(-4, 4, length=100)
hx1 <- dnorm(x1)
degf <- 3
colors <- c("red", "black")
labels <- c("df=3", "normal")

plot(x1, hx1, type="l", lty=2, xlab="x value",
  ylab="Density", main="Comparison of normal and t Distributions")

  lines(x1, dt(x1,3), lwd=2, col= "red")
legend("topright", inset=.05, title="Distributions",
  labels, lwd=2, lty=c(1, 1, 1, 1, 2), col=colors)
```



3.

```
 psi <- function(r, c = 1) {
  return(ifelse(r^2 > c^2, 2*c*abs(r) - c^2, r^2))
}
huber.loss = function(beta){
  y_hat = x %*% beta
  resid = y - y_hat
  psi = psi(r = resid ,c=1)
 result = sum(psi)
 return(result)
}
```

4

```
library("numDeriv", lib.loc="/Library/Frameworks/R.framework/Versions/3.0/Resources/library")
grad.descent <- function(f, x0, max.iter = 200, step.size = 0.001, stopping.deriv = 0.01, ...) {

  n     <- length(x0)
  xmat <- matrix(0, nrow = n, ncol = max.iter)
  xmat[,1] <- x0

  for (k in 2:max.iter) {
    # Calculate the gradient
    grad.cur <- grad(f, xmat[,k-1], ...)
    # Should we stop?
    if (all(abs(grad.cur) < stopping.deriv)) {
      k <- k-1; break
    }
    # Move in the opposite direction of the grad
    xmat[ ,k] <- xmat[ ,k-1] - step.size * grad.cur
```

```
  }

  xmat <- xmat[ ,1:k] # Trim
  return(list(x = xmat[,k], xmat = xmat, k = k))
}

x0 <- rep(0,p)
gd <- grad.descent(huber.loss,x0,max.iter = 200, step.size = 0.001, stopping.deriv = 0.1)
gd$x
```

```
## [1] -0.87346579  0.61828938  0.87989797 -0.04910821  0.07277491
## [6]  0.10229815 -0.12513246 -0.14559243 -0.11903666 -0.02250130
```

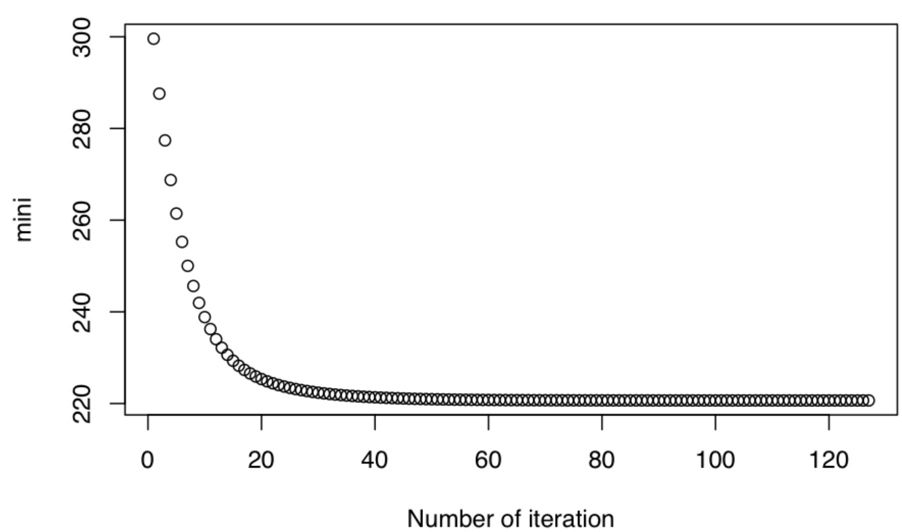
```
gd$k
```

```
## [1] 127
```

```
#There are 127 iterations.
```

5. Using gd, construct a vector obj of the values objective function encountered at each step of gradient descent.

```
mini = NULL
 for (i in 1:127){
   a = as.numeric(gd$xmat[,i])
  mini[i] =  huber.loss(beta = a )
 }
obj = c(c(1:227),mini)
plot(c(1:127),mini,xlab = "Number of iteration")
```



6.

```r
x0 <- rep(0,p)
gd1 <- grad.descent(huber.loss,x0,max.iter = 200, step.size = 0.1, stopping.deriv = 0.1)
gd1$k
```

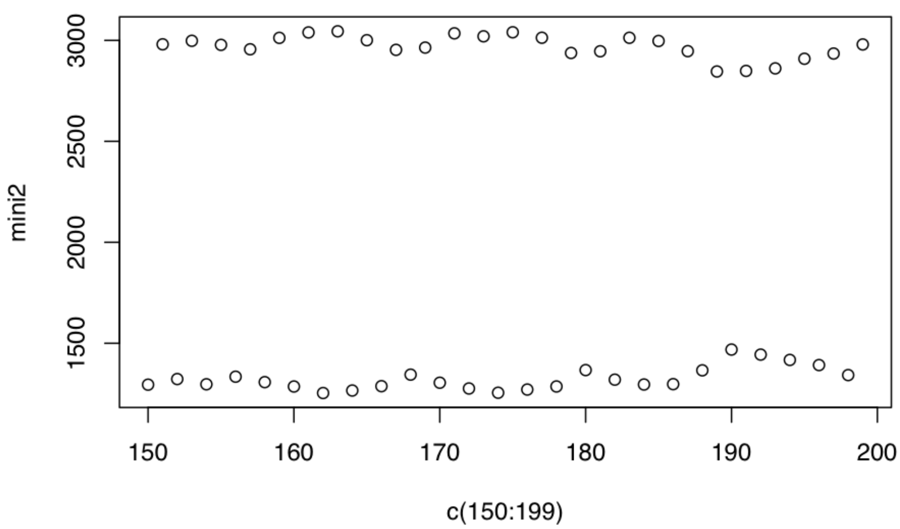
```
## [1] 200
```

```r
gd1$x
```

```
##  [1]  1.0740298 -0.7971898  2.8860325 -1.8822687  2.1897562  0.8721260
##  [7] -1.0055026 -1.5049278  0.9241456  4.7508245
```

```r
mini2 = NULL
 for (i in 1:50){
    a = as.numeric(gd1$xmat[,i+149])
  mini2[i] =  huber.loss(beta = a)
 }

plot(c(150:199),mini2)
```



7.

```r
 sparse.grad.descent = function(f, x0, max.iter = 200, step.size = 0.1, stopping.deriv = 0.05, ...) {

  n    <- length(x0)
  xmat <- matrix(0, nrow = n, ncol = max.iter)
  xmat[,1] <- x0

  for (k in 2:max.iter) {
    # Calculate the gradient
    grad.cur <- grad(f, xmat[ ,k-1],...)
```

```r
    # Should we stop?
    if (all(abs(grad.cur) < stopping.deriv)) {

      k <- k-1; break
    }

    # Move in the opposite direction of the grad
    new.x<-xmat[,k-1] - step.size * grad.cur
    new.x<-ifelse(abs(new.x) > 0.05, new.x, 0)
    xmat[ ,k] <- new.x
  }

  xmat <- xmat[ ,1:k] # Trim
  return(list(x = xmat[,k], xmat = xmat, k = k))
}
gd2 = sparse.grad.descent(huber.loss,x0,max.iter = 200, step.size = 0.001, stopping.deriv = 0.1)
gd2$k
```

```
## [1] 200
```

```r
gd2$x
```

```
##  [1] -0.8944804  0.6332991  0.8823860  0.0000000  0.0000000  0.0000000
##  [7]  0.0000000  0.0000000  0.0000000  0.0000000
```

8.

```r
coeff = coefficients(lm(y~x,data))[2:11]
mse1 = mean((b-coeff)^2);mse1
```

```
## [1] 0.1166408
```

```r
mse2 = mean((b-gd$x)^2);mse2
```

```
## [1] 0.01208955
```

```r
mse3 = mean((b-gd2$x)^2);mse3
```

```
## [1] 0.005610471
```

9.

```r
set.seed(10)
x_new <- matrix(rnorm(n*p), n, p)
y_new <- x %*% b + rt(n, df=2)
huber.loss_new <-function(beta){
  y_hat =x_new %*% beta
  resid = y_new - y_hat
  psi = psi(r = resid ,c=1)
```

```r
 result = sum(psi)
 return(result)
}
x0 <- rep(0,p)
gd3 = sparse.grad.descent(huber.loss_new,x0,max.iter = 200, step.size = 0.001, stopping.deriv = 0.1)
gd4 = grad.descent(huber.loss_new,x0,max.iter = 200, step.size = 0.001, stopping.deriv = 0.1)

mse4 = mean((b-gd3$x)^2);mse4
```

```
## [1] 0.198
```

```r
mse5 = mean((b-gd4$x)^2);mse5
```

```
## [1] 0.352788
```

10.

```r
mse_gd = NULL
mse_sgd = NULL
for (i in 1:10){
x0 <- rep(0,p)
gd_10 <- grad.descent(huber.loss,x0,max.iter = 200, step.size = 0.001, stopping.deriv = 0.1)
mse_gd[i] = mean((b-gd_10$x)^2)
sparse_gd_10 = sparse.grad.descent(huber.loss,x0,max.iter = 200, step.size = 0.001, stopping.deriv = 0.:
mse_sgd[i] =  mean((b-sparse_gd_10$x)^2)
}
mean_mse_gd = mean(mse_gd);mean_mse_gd
```

```
## [1] 0.01208955
```

```r
mean_mse_sgd = mean(mse_sgd);mean_mse_sgd
```

```
## [1] 0.005610471
```

```r
min_mse_gd = min(mse_gd);min_mse_gd
```

```
## [1] 0.01208955
```

```r
min_mse_sgd = min(mse_sgd);min_mse_sgd
```

```
## [1] 0.005610471
```