STAT 580 HW6

Haozhe Zhang

April 18, 2017

Problem 1

(a)

$$\ell(\theta) = \sum_{i=1}^{n} log\{p(x_i - \theta)\} = -nlog\pi - \sum_{i=1}^{n} log\{1 + (\theta - x_i)^2\}$$

$$\ell'(\theta) = -2\sum_{i=1}^{n} \frac{\theta - x_i}{1 + (\theta - x_i)^2}$$

$$\ell''(\theta) = -2\sum_{i=1}^{n} \frac{1 - (\theta - x_i)^2}{\{1 + (\theta - x_i)^2\}^2}$$

(b)

$$I(\theta) = -E\{\ell''(\theta)\} = \frac{2n}{\pi}\int \frac{1 - x^2}{(1 + x^2)^3} dx = \frac{2n}{\pi} \times \frac{\pi}{4} = \frac{n}{2}$$

(c)

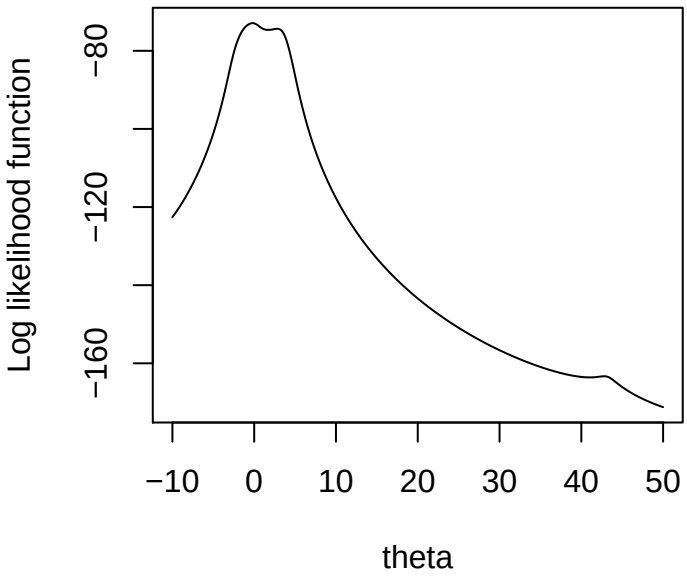
```
dat <- c(-13.87,-2.53,-2.44,-2.4,-1.75,
         -1.34,-1.05,-0.23,-0.07,0.27,1.77,
         2.76,3.29,3.47,3.71,3.8,4.24,4.53,
         43.21,56.75)

loglik <- function(theta){
  -length(dat)*log(pi) - sum(log(1+(theta-dat)^2))
}

loglik_deriv1 <- function(theta){
  -2*sum((theta-dat)/(1+(theta-dat)^2))
}

loglik_deriv2 <- function(theta){
  -2*sum((1-(theta-dat)^2)/(1+(theta-dat)^2)^2)
}

plot(seq(-10,50,0.01),sapply(seq(-10,50,0.01),loglik),
     xlab="theta", ylab="Log likelihood function", type="l")
```



(d)

```
Newton_Raphson <- function(init_theta, thres_error){
  theta_0 <- init_theta
  ratio <- thres_error+1
  while(ratio > thres_error){
    theta_1 <- theta_0 - loglik_deriv1(theta_0)/loglik_deriv2(theta_0)
    ratio <- abs((theta_1-theta_0)/(theta_0+10^(-8)))
    theta_0 <- theta_1
    if(abs(theta_0)>10^(10)) return("Not convergent")
  }
  return(theta_1)
}

Newton_Raphson(11,10^(-8))
```

```
## [1] 54.87662
```

```
Newton_Raphson(1,10^(-8))
```

```
## [1] 1.713587
```

```
Newton_Raphson(0,10^(-8))
```

```
## [1] -0.1922866
```

```
Newton_Raphson(1.4,10^(-8))
```

```
## [1] 1.713587
```

```
Newton_Raphson(4.1,10^(-8))
```

```
## [1] 2.817472
```

```
Newton_Raphson(4.8, 10^(-8))
```

```
## [1] "Not convergent"
```

```
Newton_Raphson(7,10^(-8))
```

```
## [1] 41.04085
```

```
Newton_Raphson(8,10^(-8))
```

```
## [1] "Not convergent"
```

```
Newton_Raphson(38,10^(-8))
```

```
## [1] 42.79538
```

(e)

Comparison: for two starting points (4.8, 8), Newton-Raphson method doesn't converge, while Fisher scoring method converges for all the starting points.

```
Newton_Scoring <- function(init_theta, thres_error){
  theta_0 <- init_theta
  ratio <- thres_error+1
  while(ratio > thres_error){
    theta_1 <- theta_0 - loglik_deriv1(theta_0)/(length(dat)/2)
    ratio <- abs((theta_1-theta_0)/(theta_0+10^(-8)))
    theta_0 <- theta_1
    if(abs(theta_0)>10^(10)) return("Not convergent")
  }
  Newton_Raphson(theta_1,thres_error)
}

Newton_Scoring(11,10^(-8))
```

```
## [1] 41.04085
```

```
Newton_Scoring(1,10^(-8))
```

```
## [1] 1.713587
```

```
Newton_Scoring(0,10^(-8))
```

```
## [1] 1.713587
```

```
Newton_Scoring(1.4,10^(-8))
```

```
## [1] 1.713587
```

```
Newton_Scoring(4.1,10^(-8))
```

```
## [1] 41.04085
```

```
Newton_Scoring(4.8,10^(-8))
```

```
## [1] 41.04085
```

```
Newton_Scoring(7,10^(-8))
```

```
## [1] 41.04085
```

```
Newton_Scoring(8,10^(-8))
```

```
## [1] 41.04085
```

```
Newton_Scoring(38,10^(-8))
```

```
## [1] 41.04085
```

Problem 2

(a)

```
dat <- data.frame(x=c(0.02,0.06,0.11,0.22,0.56,1.1),
                  y=c(47/76,97/107,123/139,152/159,191/201,200/207))
1/lm(y~x,data=1/dat)$coefficients[1]
```

```
## (Intercept)
##   0.9991668
```

```
lm(y~x,data=1/dat)$coefficients[2]/lm(y~x,data=1/dat)$coefficients[1]
```

```
##          x
## 0.01178085
```

(b) Newton-Raphson algorithm

```
g <- function(theta){
  sum((dat$y-theta[1]*dat$x/(dat$x+theta[2]))^2)
}

g_deriv1 <- function(theta){
  a1 <- -2*sum((dat$y-theta[1]*dat$x/(dat$x+theta[2]))*dat$x/(dat$x+theta[2]))
  a2 <- 2*sum((dat$y-theta[1]*dat$x/(dat$x+theta[2]))*theta[1]*dat$x/(dat$x+theta[2])^2)
  return(matrix(c(a1,a2),2,1))
}

g_deriv2 <- function(theta){
  a11 <- sum(2*(dat$x/(dat$x+theta[2]))^2)
  a12 <- sum(2*dat$x*dat$y/(dat$x+theta[2])^2-4*theta[1]*dat$x^2/(dat$x+theta[2])^3)
  a22 <- sum(6*dat$x^2*theta[1]^2/(dat$x+theta[2])^4-4*theta[1]*dat$x*dat$y/(dat$x+theta[2])^3)
  return(matrix(c(a11,a12,a12,a22),2,2))
}

Newton_Raphson <- function(init_theta, thres_error){
  theta_old <- matrix(init_theta,2,1)
  ratio <- thres_error + 1
  while(ratio > thres_error){
    if(min(abs(eigen(g_deriv2(theta_old))$values)) < 10^(-5))
      theta_new <- theta_old - solve(g_deriv2(theta_old)+diag(10^(-5),2))%*% g_deriv1(theta_old)
    else
      theta_new <- theta_old - solve(g_deriv2(theta_old))%*% g_deriv1(theta_old)

    ratio <- sum((theta_new - theta_old)^2)/(sum(theta_old^2)+10^(-10))
    theta_old <- theta_new
  }
```

```
  return(theta_new)
}

Newton_Raphson(c(0.9991668,0.01178085), 10^(-5))
```

```
##            [,1]
## [1,] 0.98790881
## [2,] 0.01043724
```

(c) Steepest descent algorithm.

```
Steepest_Descent <- function(init_theta, n_iteration, init_alpha){
  theta_old <- matrix(init_theta,2,1)
  alpha <- init_alpha
  iter <- 1
  while(iter<n_iteration){
    theta_new <- theta_old - alpha*g_deriv1(theta_old)
    if(g(theta_new) > g(theta_new)){
      alpha <- alpha/2
    }
    else{
      theta_old <- theta_new
      iter <- iter + 1
    }
  }
  return(theta_old)
}

Steepest_Descent(c(0.9991668,0.01178085),10000,0.0001)
```

```
##            [,1]
## [1,] 0.98795603
## [2,] 0.01044074
```

(d) Gauss-Newton algorithm

```
A <- function(theta){
  cbind(-2*(dat$y-theta[1]*dat$x/(dat$x+theta[2]))*dat$x/(dat$x+theta[2]),
        2*(dat$y-theta[1]*dat$x/(dat$x+theta[2]))*theta[1]*dat$x/(dat$x+theta[2])^2)
}

Z <- function(theta){
  matrix(dat$y - (dat$y - theta[1]*dat$x/(dat$x+theta[2]))^2,length(dat$x),1)
}

Gauss_Newton <- function(init_theta, thres_error){
  theta_old <- matrix(init_theta,2,1)
  ratio <- thres_error + 1
  while(ratio > thres_error){
    if(min(abs(eigen(t(A(theta_old))%*%A(theta_old))$values)) < 10^(-5))
      theta_new <- theta_old + solve(t(A(theta_old))%*%A(theta_old)+diag(10^(-5),2))%*%
```

```
        t(A(theta_old))%*%Z(theta_old)
    else
      theta_new <- theta_old + solve(t(A(theta_old))%*%A(theta_old))%*%
      t(A(theta_old))%*%Z(theta_old)
    ratio <- sum((theta_new - theta_old)^2)/(sum(theta_old^2)+10^(-10))
    theta_old <- theta_new
    theta_new
  }
  return(theta_new)
}
init_theta <- c(1,0.01);
Gauss_Newton(init_theta,0.000001)
```

```
##              [,1]
## [1,] 1.976206605
## [2,] 0.007275901
```