STATS 406F15 Lab 11

1 Optimization using the optim function

Problem 1

Suppose $X_1, \dots, X_n$ are i.i.d. $N(\mu, \sigma^2)$, where both μ and σ^2 are unknown. We will compute the MLE for parameters (μ, σ^2). The log-likelihood is:

$$l(\mu, \sigma^2) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}(n\log(\sigma^2) + \frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2) \ .$$

Generate data with $\mu = 2$ and $\sigma^2 = 4$.

Solution

```
## Simulate 100 data points from Normal distribution with mu=2, sigma^2=4
X <- rnorm(100, mean=2, sd=sqrt(4))
n <- 100

# likelihood t[1]: mu, t[2]: sigma^2
# if sigma < 0 return Inf, instead of NaN.
f <- function(t)
{print(X)
    if( t[2] > 0)
    {
        return( -sum( dnorm(X, mean=t[1], sd=sqrt(t[2]), log=TRUE) ) )
    } else
    {
        return(Inf)
    }
}

# gradient function
df <- function(t)
{
    mu <- t[1]; sig <- t[2];
    g <- rep(0,2)
    g[1] <- (1/sig) * sum(X - mu)
    g[2] <- (-n/(2*sig)) + sum( (X-mu)^2 )/(2*sig^2)
    return(-g)
}

## Run optim()
objoptim <- optim(par=c(0, 1), fn=f, gr=df, method='BFGS')
print(objoptim$par)
```

2 Newton-Raphson methods

Newton's method for solving $f(x) = 0$

$$y = f(x_0) + f'(x_0)(x - x_0)$$

The tangent line crosses 0 when $y = 0$, which happens when

$$f(x_0) = -f'(x_0)(x - x_0)$$

rearranging terms approporiately, we find that

$$x = x_0 - f(x_0)/f'(x_0)$$

Now setting $x_0 = x$ and repeating until convergence is essentially Newton s method. We can see graphically how this works with the function: $f(x) = e^x - 5$

Solution

```
# f and f'
f <- function(x) exp(x)-5
df <- function(x) exp(x)
# make the initial plot
v <- seq(0,4,length=1000)
plot(v,f(v),type="l",col=8)
abline(h=0)
# start point
x0 <- 3.5
# paste this repeatedly to see Newton's method work
segments(x0,0,x0,f(x0),col=2,lty=3)
slope <- df(x0)
g <- function(x) f(x0) + slope*(x - x0)
v = seq(x0 - f(x0)/df(x0),x0,length=1000)
```

```
lines(v,g(v),lty=3,col=4)
x0 <- x0 - f(x0)/df(x0)
```

Newton's method for finding where $f(x)$ is at its maximum

We can write this as a generic function in R as follows.

```
# f: the function you want the max of
# df: derivative of f; d2f: second derivative
# x0: start value, tol: convergence criterion
# maxit: max # of iterations before it is considered to have failed
newton <- function(x0, f, df, d2f, tol=1e-4, pr=FALSE){
  # iteration counter
  k <- 0
  # initial function, derivatives, and x values
  fval <- f(x0)
  grad <- df(x0)
  hess <- d2f(x0)
  xk_1 <- x0
  cond1 <- sqrt(sum(grad^2))
  cond2 <- Inf
  # see if the starting value is already close enough
  if( (cond1 < tol) ) return(x0)
  while( (cond1 > tol) & (cond2 > tol) )
  {
    L <- 1
    bool <- TRUE
    while(bool == TRUE){
      xk <- xk_1 - L * solve(hess) %*% grad
      # see if we've found an uphill step
      if( f(xk) > fval ){
        bool = FALSE
        grad <- df(xk)
        fval <- f(xk)
        hess <- d2f(xk)
        # make the stepsize a little smaller
      }
      else {
        L = L/2
        if( abs(L) < 1e-20 ) return("Failed to find uphill step - try new start val")
      }
    }
    # calculate convergence criteria
    cond1 <- sqrt( sum(grad^2) )
    cond2 <- sqrt( sum( (xk-xk_1)^2 ))/(tol + sqrt(sum(xk^2)))
    # add to counter and update x
    k <- k + 1
    xk_1 <- xk
  }
  if(pr == TRUE) print( sprintf("Took %i iterations", k) )
  return(xk)
}
```

Problem 2

Let $f(x) = e^{-(x^2)}$. It is easy to see that $f'(x) = -2xf(x)$ and $f'(x) = -2f(x) - 2xf'(x)$. Now we code each of these functions and call the newton () function:

Solution

```
f <- function(x) exp(-(x^2))
df <- function(x) -2*x*f(x)
d2f <- function(x) -2*f(x)-2*x*df(x)
# Newton's method starting at x0 = 2/3
newton(2/3, f, df, d2f, 1e-7)
```

The maximum at 0 appears to have been found.

Problem 3

Suppose $X_1, \ldots, X_n$ are iid $N(\mu,\ \sigma^2)$ where both μ and σ^2 are unknown. We will write a program to do maximum likelihood estimation for $\theta = (\mu,\ \sigma^2)$. The log-likelihood is

$$l(\mu,\ \sigma^2) = -\frac{1}{2}[n\ \log(2\pi\sigma^2) + \frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2]$$

so the elements of the gradient are

$$\frac{\partial l}{\partial \mu} = \frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i - \mu)$$

and

$$\frac{\partial l}{\partial \sigma^2} = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{i=1}^{n}(x_i - \mu)^2$$

finally, the elements of the hessian are

$$\frac{\partial^2 l}{\partial \mu^2} = -\frac{n}{\sigma^2},$$

$$\frac{\partial^2 l}{\partial(\sigma^2)^2} = \frac{n}{2\sigma^4} - \frac{1}{\sigma^6}\sum_{i=1}^{n}(x_i - \mu)^2$$

and

$$\frac{\partial^2 l}{\partial\mu\partial\sigma^2} = -\frac{1}{\sigma^4}\sum_{i=1}^{n}(x_i - \mu)$$

We can code these functions easily in R and call newton to get the MLEs

```
X <- rnorm(100, mean=2, sd=sqrt(4))
n <- 100
# likelihood t[1]: mu, t[2]: sigma^2
# if sigma < 0 return -Inf, instead of NaN.
f <- function(t){
  if( t[2] > 0)
  { return( sum( dnorm(X, mean=t[1], sd=sqrt(t[2]), log=TRUE) ) )
  }
  else {
  return(-Inf)
  }
}
# score function
df <- function(t){
  mu <- t[1]; sig <- t[2];
  g <- rep(0,2)
  g[1] <- (1/sig) * sum(X - mu)
  g[2] <- (-n/(2*sig)) + sum( (X-mu)^2 )/(2*sig^2)
  return(g)
}
# hessian
d2f <- function(t){
  mu <- t[1]; sig <- t[2];
  h <- matrix(0,2,2)
  h[1,1] <- -n/sig
  h[2,2] <- (n/(2*sig^2)) - sum( (X-mu)^2 )/(sig^3)
  h[1,2] <- -sum( (X-mu) )/(sig^2)
  h[2,1] <- h[1,2]
  return(h)
}
newton( c(0,1), f, df, d2f)
```

3 EM algorithm

Example

Consider a simple regression model $Y_i = \beta x_i + \epsilon_i$, where $\epsilon_i \sim N(0,\ 1)$, for $i = 1 \dots n+m$. Suppose that the first n data Y_i are observed while the last m are censored at some positive threshold C. Censoring means that all we know is that $Y_i > C$, but we do not actually observe Y_i. We wish to estimate β. If we have seen all the Y_i, then the complete likelihood is

$$l_{com}(\beta|Y) = -\frac{\beta^2}{2}\sum_{i=1}^{n+m} x_i^2 + \beta\sum_{i=1}^{n+m} x_i y_i + const.$$

Therefore the estimate of β in complete data setting is $[\sum\limits_{i=1}^{n+m} x_i y_i]/[\sum\limits_{i=1}^{n+m} x_i^2]$. With censoring, all we see on the last m samples is that $I_i = 1$, where $I_i = 1$ if $Y_i > C$, and $I_i = 0$ otherwise. In this problem the complete likelihood depends only linearly on Y_i. So to implement the EM, we only need to find $E(Y_i|I_i = 1)$, for a given value of β. Since $Z_i = Y_i - x_i\beta \sim N(0,1)$, we have $E(Y_i|I_i = 1) = x_i\beta + E(Z_i|I_i = 1)$. Further

$$E(Z_i|I_i = 1) = E(Y_i - x_i\beta | Y_i - x_i\beta > C - x_i\beta) = E(Z_i|Z_i > C - x_i\beta) = \frac{\int_{C-x_i\beta}^{\infty} z\phi(z)dz}{1-\Phi(C - x_i\beta)}$$

$$E(Y_i|I_i = 1) = x_i\beta + \frac{\phi(C - x_i\beta)}{1-\Phi(C - x_i\beta)} = m_i(\beta).$$

Given β_k, if we replace the Y_i for $n+1 \le i \le n+m$ by $m_i(\beta_k)$, we obtain the Q function

$$Q(\beta|\beta_k) = -\frac{\beta^2}{2}\sum_{i=1}^{n+m} x_i^2 + \beta\sum_{i=1}^{n} x_i y_i + \beta\sum_{i=n+1}^{n+m} x_i m_i + const.$$

Maximizing this function gives easily the solution

$$\frac{\sum_{i=1}^{n} x_i y_i + \sum_{i=n+1}^{n+m} x_i m_i}{\sum_{i=1}^{n+m} x_i^2}$$

We obtain the following EM algorithm for estimating the mle of β.

Algorithm (The EM algorithm for the censored data model).

(E step): At stage k, given β_k a current guess of β, compute $m_i = x_i\beta_k + \phi(C - x_i\beta_k)/[1 - \Phi(C - x_i\beta_k)]$, for $i = (n+1),\ \dots,\ (n+m)$.

(M step): Compute the new estimate of β,

$$\beta_{k+1} = \frac{\sum_{i=1}^{n} x_i y_i + \sum_{i=n+1}^{n+m} x_i m_i}{\sum_{i=1}^{n+m} x_i^2}$$

Solution

```
############
beta_star=2; n_0=100
X=rnorm(n_0,0,1)
Y=rnorm(n_0,mean=beta_star*X,sd=1)
C=2.5
n=sum(Y<=C); m=sum(Y>C)
X=c(X[Y<=C],X[Y>C])
Y=Y[Y<=C]
beta=0
K=50
Res=double(K)
for (k in 1:K){
  m_beta= X[(n+1):(n+m)]*beta
  + dnorm(C-X[(n+1):(n+m)]*beta)/(1-pnorm(C-X[(n+1):(n+m)]*beta))
  beta =( sum(X[1:n]*Y) + sum(X[(n+1):(n+m)]*m_beta) ) / ( sum(X^2) )
  Res[k]=beta
}
plot(Res,type='b',col='blue')
```