STA 250: HW3: OPTIMIZATION MODULE

CHRISTOPHER CONLEY

1. Question 1: Bisection & Newton-Raphson

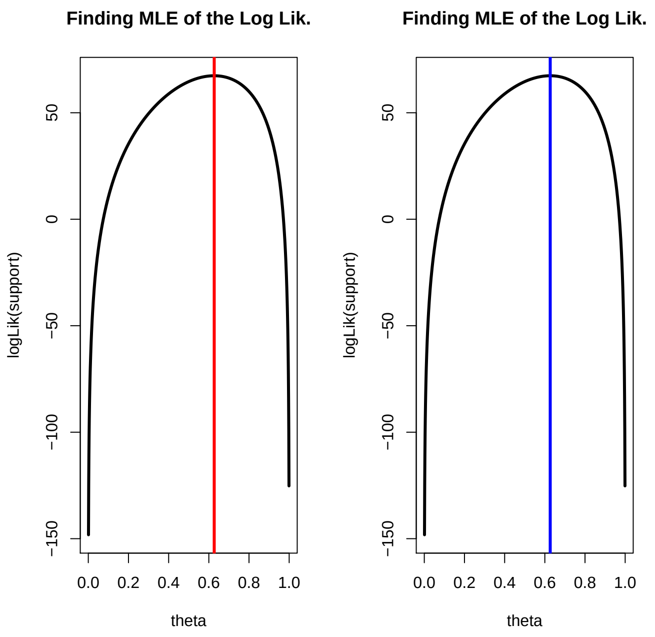

Figure 1. The log likelihood (y axis) is maximized with negligible differing error at $\hat{\theta} = 0.6268215$ for both bisection (red) and Newton-Raphson (blue).

1.1. Bisection.

```
bisection <- function(a, b, g, epsilon, maxIter, verbose = FALSE) {
```

```
#constraints on the paramters
stopifnot(g(a) * g(b) < 0,
          a < b,
          epsilon > 0,
          is.function(g),
          is.integer(maxIter))

#initialize
lower <- a
upper <- b
converged <- FALSE
iterCount <- 0

while (!converged && iterCount < maxIter) {



  if (verbose == TRUE) {
    cat("iteration: ", iterCount, "\n");
    cat("root (current): ", center, "\n");
  }

  iterCount <- iterCount + 1
  center <- (lower + upper) / 2
  if ( abs(g(center)) < epsilon) {
    converged <- TRUE
  } else {
    if ( g(lower) * g(center) < 0) {
      upper <- center
    } else {
        #g(center) * g(upper) < 0
        lower <- center
      }
    }
  }
  return(center)
}
```

1.2. Newton-Raphson.

```
newtonRaphson <- function(g, gprime, xInit, epsilon, maxIter, verbose = FALSE) {

  #constraints on the parameters
  stopifnot(is.function(g),
            is.function(gprime),
            epsilon > 0,
            is.integer(maxIter))

  #small update function
  eta <- function(u, g, gprime) {
    - g(u)  / gprime (u)
  }

  #initalize key variables
  x <- xInit
  converged <- FALSE
  iterCount <- 0
  while(!converged && iterCount < maxIter) {

    if (verbose == TRUE) {
      cat("iteration: ", iterCount, "\n");
      cat("root (current): ", x, "\n");
    }
    iterCount <- iterCount + 1

    if ( abs(g(x))< epsilon ) {
      converged <- TRUE
    }

    x <- x + eta(x, g, gprime)
  }
  return(x)
}
```

1.3. Linkage Problem.

```
setwd("~/myrepos/sta250/Stuff/HW3/")

#genotype frequencies
countAB <- 125
countAb <- 18
countaB <- 20
countab <- 34
```

```
lambdaFunc <- function(theta) {
  1 - 2*theta + theta*theta
}

#log likelihood
logLik <- function(lambda) {
  countAB*log( 2 + lambda) + (countAb + countaB)*log (1 - lambda) + countab*log(lambda)
}

#first derivative of the log likelihood
logLikPrime <- function(lambda) {
  (countAB / (2 + lambda)) - ((countAb + countaB) / (1 - lambda)) + (countab / lambda)
}

#first derivative of the log likelihood
logLikDoublePrime <- function(lambda) {
  (-countAB / (2 + lambda)^2) - ((countAb + countaB) / (1 - lambda)^2) - (countab / lambda^2)
}



#find a good starting value for either algorithm
#0 <= theta <= 1
large.lambda <- lambdaFunc(0)
small.lambda <- lambdaFunc(1)
support <- seq(from = 0, to = 1, length = 1000)
plot(support, lambdaFunc(support), type = 'l')

#numerical accuracy
```

```
epsilon <- 1e-10
maxIter <- as.integer(1e5)

source("bisection.r")
lambdaBisection <- bisection(a = small.lambda, b = large.lambda,
                             g = logLikPrime, epsilon = epsilon,
                             maxIter = maxIter)
lambdaBisection

source("newton-raphson.r")
lambdaNR <- newtonRaphson(g = logLikPrime, gprime  = logLikDoublePrime,
                          xInit = 0.5, epsilon = epsilon, maxIter)
lambdaNR

pdf("log-lik-mle.pdf")
par(mfrow = c(1,2))
plot(support, logLik(support), type = 'l', lwd = 3,
     main = "Finding MLE of the Log Lik.", xlab = "theta")
abline(v = lambdaBisection, col = "red", lwd = 3)

plot(support, logLik(support), type = 'l', lwd = 3,
     main = "Finding MLE of the Log Lik.", xlab = "theta")
abline(v = lambdaNR, col = "blue", lwd = 3)
dev.off()
```