STAT:5400
Lecture 14
Oct. 8, 2018
Root-finding

Root finding algorithms

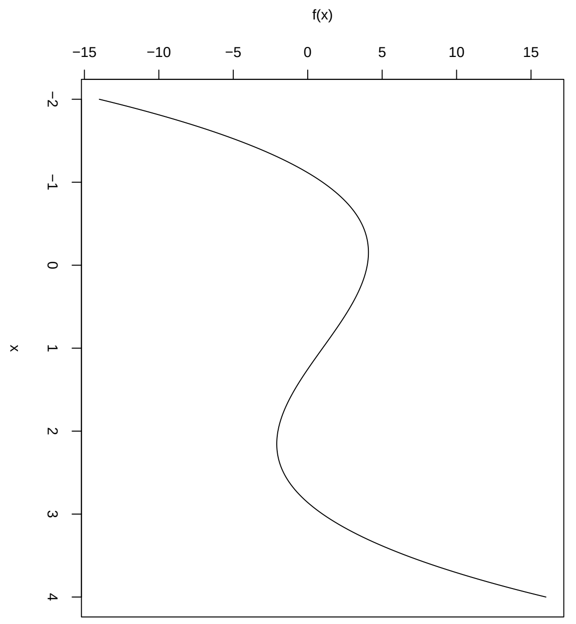
- problem: to find values of variable x that satisfy $f(x) = 0$ for given function f
- solution is called "zero of f" or "root of f"
- when is this an important problem in statistics?

The bisection method

- also called "binary-search method"
- conditions for use
  - f continuous, defined on interval $[a, b]$
  - $f(a)$ and $f(b)$ of opposite sign
- by Intermediate Value Theorem, there exists a $p,\ a < p < b$, such that $f(p) = 0$
- procedure works when $f(a)$ and $f(b)$ of opposite sign and more than one root in $[a, b]$
- for simplicity, we'll assume unique root in interval
- method consists of
  - repeated halving of subintervals of $[a, b]$
  - at each step, locating half containing p
- requires following inputs
  - endpoints a, b
  - tolerance TOL
  - maximum number of iterations N_0


```
function(func, a, b, tol, maxiters)
{

# bisection
# uses bisection algorithm to find root of func in interval [a,b]
# Burden and Faires, section 2.1

################################################################
# inputs
################################################################
# tol      -- maximum difference between subinterval endpoints to
#             consider root to have been found
# f        -- function for which root needs to be found
# a,b      -- interval endpoints, b > a
# maxiters -- maximum number of iterations
################################################################

# initial setup

if( f(a) * f(b) > 0)
print("Function has same sign at both endpoints.")
else {
absdiff <- b-a
iters <- 1
        p <- a + absdiff / 2
while ((absdiff > tol) & (iters <= maxiters) & (f(p) != 0) )
         {
         absdiff <- b-a # note: absdiff is constructed to be positive
         p <- a + absdiff / 2
         if ((f(p) != 0) && (absdiff > tol)) {
if( f(p) * f(a) < 0 )
b <- p
else
a <- p
iters <- iters + 1
}
}
if (iters > maxiters )   # didn't find solution in fewer than maxiters
print("Maximum number of iterations exceeded.")

list( a = a, b = b, p = p, errflag = as.numeric(iters > maxiters) )
}


}
```

Example




```
> f <- function(x) {x^3 - 3*x^2 -x + 4}

> bisection( f, -2,4, .0001, 100 )
$a
[1] -1.114960

$b
[1] -1.114868

$p
[1] -1.114914

$errflag
[1] 0
```


uniroot function in R

```
uniroot                 package:stats                 R Documentation

One Dimensional Root (Zero) Finding

Description:

     The function 'uniroot' searches the interval from 'lower' to
     'upper' for a root (i.e., zero) of the function 'f' with respect
     to its first argument.

Usage:

     uniroot(f, interval, lower = min(interval), upper = max(interval),
             tol = .Machine$double.eps^0.25, maxiter = 1000, ...)
```



```
> f <- function(x) {x^3 - 3*x^2 -x + 4)
> plot(seq(-2,4,by=0.01), f(seq(-2,4,by=0.01)),type="l")
> uniroot(f=f,c(-2,4))
$root
[1] -1.114907

$f.root
[1] 9.607438e-06

$iter
[1] 9

$estim.prec
[1] 6.103516e-05
```

The Newton-Raphson Method

- one of most powerful and well-known numerical methods for solving root-finding problem $f(x) = 0$
- one derivation: Taylor series approximation
  - suppose f' and f'' are continuous on $[a, b]$
  - let $x_0 \in [a, b]$ be an approximation to p such that $f'(x_0) \neq 0$ and $|x_0 - p|$ is "small"
  - first order Taylor approximation for $f(x)$ expanded around x_0

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{(x - x_0)^2}{2} f''(\xi(x))$$

    where $\xi(x)$ is between x and x_0.
  - with $x = p$ this gives

$$0 = f(x_0) + (p - x_0)f'(x_0) + \frac{(p - x_0)^2}{2} f''(\xi(x))$$

  - since $|x_0 - p|$ is "small", $(x_0 - p)^2$ should be negligible and

$$0 \simeq f(x_0) + (p - x_0)f'(x_0)$$

  - solving for p yields

$$p \simeq x_0 - \frac{f(x_0)}{f'(x_0)}$$

The Newton-Raphson Method

- start with initial approximation p_0
- let $p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}$

`http://en.wikipedia.org/wiki/File:NewtonIteration_Ani.gif`

Convergence Theorem for Newton-Raphson Method

- conditions
  - f has continuous first and second derivatives on $[a, b]$
  - $p \in [a, b]$ is such that $f(p) = 0$ and $f'(p) \neq 0$
- conclusions
  - then there exists a $\delta > 0$ such that Newton's method generates a sequence $\{p_n\}_{n=1}^{\infty}$ converging to p for any initial approximation $p_0 \in [p - \delta, p + \delta]$.

The Secant Algorithm

- useful when computation of $f'(x)$ is far more computationally intensive than computation of $f(x)$
- uses forward (or backward)-difference formula to approximate $f'(p_{n-1})$

$$f'(p_{n-1}) \simeq \frac{f(p_{n-2}) - f(p_{n-1})}{p_{n-2} - p_{n-1}} = \frac{f(p_{n-1}) - f(p_{n-2})}{p_{n-1} - p_{n-2}}$$

- Secant algorithm generates sequence as

$$p_n = p_{n-1} - \frac{f(p_{n-1})(p_{n-1} - p_{n-2})}{f(p_{n-1}) - f(p_{n-2})},\ n \geq 1$$