Bisection, Newton, and secant methods

Source: Ahmadi, ORF 363 slides.

Bisection method

Newton method

Secant method

- Have you ever looked for a book in a library?
- Or searched for a word in the dictionary?
- If you have, then you already know bisection!

Suppose

- $f : \mathbb{R} \to \mathbb{R}$ is continuously differentiable
- We know interval $[a_0, b_0]$ which has only one local minimum
- The derivative of changes sign only once on $[a_0, b_0]$

Bisection algorithm

Set a small $\epsilon > 0$

For $k = 0 : N$
  if $\left| f'\left(\frac{a_k + b_k}{2}\right) - \epsilon \right| > 0$
    you are done
  elseif $f'\left(\frac{a_k + b_k}{2}\right) \geq \epsilon$
    $a_{k+1} = a_k$
    $b_{k+1} = \frac{a_k + b_k}{2}$
  elseif $f'\left(\frac{a_k + b_k}{2}\right) \leq \epsilon$
    $a_{k+1} = a_k$
    $b_{k+1} = \frac{a_k + b_k}{2}$
  end
end

Remarks

- In each iteration the length of the interval where the minimum lies is cut in half; i.e.

$$|a_{k+1} - b_{k+1}| = \frac{1}{2}|a_k - b_k|$$

Remarks

- In each iteration the length of the interval where the minimum lies is cut in half; i.e.

$$|a_{k+1} - b_{k+1}| = \frac{1}{2}|a_k - b_k|$$

- The number of steps that we take is

$$N \leq \log_2 \frac{|a_0 - b_0|}{\epsilon}$$

Remarks

- In each iteration the length of the interval where the minimum lies is cut in half; i.e.

$$|a_{k+1} - b_{k+1}| = \frac{1}{2}|a_k - b_k|$$

- The number of steps that we take is

$$N \leq \log_2 \frac{|a_0 - b_0|}{\epsilon}$$

- Bisection not as fast as the next two algorithms we'll see

Remarks

- In many applications, we know a priori an initial interval $[a_0.b_0]$
- But what if we don't? The following strategy can do the job
  - Find 3 points $a < c < b$ such that

$$f(c) < f(a) \text{ and } f(c) < f(b)$$

  - Set initial inverval as $[a, b]$

A common use of bisection

Consider an optimization problem:

$$\begin{aligned} \min \;\; & f(x) \\ s.t. \;\; & g_i(x) \leq 0 \end{aligned}$$

Suppose we have a black box that can test for feasibility – it tells us whether the set $\{x \mid g_i(x) \leq 0\}$ is empty or not

A common use of bisection

Our problem is equivalent to:

$$\begin{aligned} \min \ & \gamma \\ s.t. \ & g_i(x) \leq 0, f(x) \leq \gamma \end{aligned}$$

- We can do bisection on γ. For each fixedγ, call the feasibility black box on the set $\{g_i(x) \leq 0, f(x) \leq \gamma\}$
  - If feasible, decrease γ
  - If infeasible, increase γ

.

Bisection method

Newton method

Secant method

Newton's method

Suppose that we have access to f' and f''. Newton's method (aka the Newton-Raphson method) for minimization is based on the following iterations:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

Derivation of Newton's method

Let $g(x) = f'(x)$. We want to solve $g(x) = 0$

Comparison between bisection and Newton

Finding a minimizer of $f(x) = \frac{x^4}{4} - 10x \Rightarrow f'(x) = x^3 - 10$

Answer = 2.154434690031884

	Newton	Bisection		
k	x_k	a_k	$\frac{a_k+b_k}{2}$	b_k
1.000000000000000	12.000000000000000	0	6.000000000000000	12.000000000000000
2.000000000000000	8.023148148148149	0	3.000000000000000	6.000000000000000
3.000000000000000	5.400548660419450	0	1.500000000000000	3.000000000000000
4.000000000000000	3.714654390828676	1.500000000000000	2.250000000000000	3.000000000000000
5.000000000000000	2.718005659267038	1.500000000000000	1.875000000000000	2.250000000000000
6.000000000000000	2.263213061967483	1.875000000000000	2.062500000000000	2.250000000000000
7.000000000000000	2.159579216386407	2.062500000000000	2.156250000000000	2.250000000000000
8.000000000000000	2.154446935535738	2.062500000000000	2.109375000000000	2.156250000000000
9.000000000000000	2.154434690101485	2.109375000000000	2.132812500000000	2.156250000000000
10.000000000000000	2.154434690031884	2.132812500000000	2.144531250000000	2.156250000000000
11.000000000000000	2.154434690031884	2.144531250000000	2.150390625000000	2.156250000000000

Bad cases for Newton's method

1. Moved in the wrong direction

Bad cases for Newton’s method

2. Iterations diverge

Bad cases for Newton's method

3. Convergence can get slow if $g'(x^*) = f''(x^*) = 0$

Bisection method

Newton method

Secant method

Secant method

Recall the Newton's method:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

In the case that you don't want to compute $f''(x)$, for example, $f''(x) = e^{-(x-2)^2}$, you can approximate it with **finite difference**:

$$f''(x_k) \approx \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}}$$

Secant method

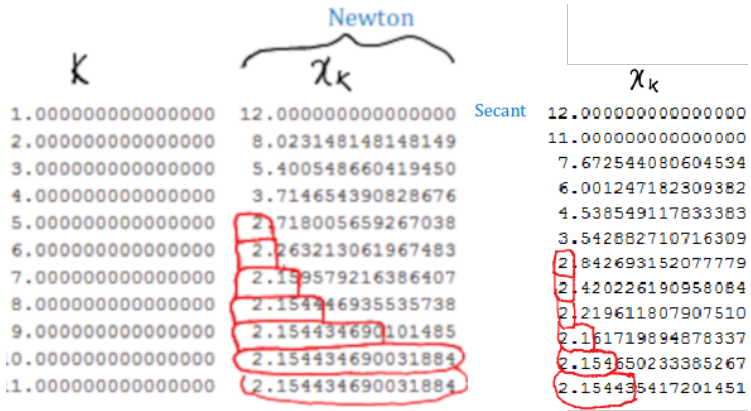
The **Secant method**:

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f'(x_k) - f'(x_{k-1})} f'(x_k)$$

Comparison between Newton's and Secant methods

Finding a minimizer of $f(x) = \frac{x^4}{4} - 10x \Rightarrow f'(x) = x^3 - 10$

Answer = 2.154434690031884



Remarks

- We have been optimizing f via **root finding** i.e. solving for $f'(x) = 0$
- An x^* that satisfies $g(x^*) = 0$ is called a **root** of g

Remarks

- We have been optimizing f via **root finding** i.e. solving for $f'(x) = 0$
- An x^* that satisfies $g(x^*) = 0$ is called a **root** of g
- We have done it in one dimension. How hard is it in higher dimensions?

Remarks

- Finding **real roots** in many dimensions
  - Set of linear equations? Can be done efficiently (polynomial time)

$$N \leq \text{Polynomial}(\text{dimension},\ \#\ \text{of equations})$$

Remarks

- Finding **real roots** in many dimensions
  - Set of linear equations? Can be done efficiently (polynomial time)

$$N \leq \text{Polynomial}(\text{dimension},\ \#\ \text{of equations})$$

  - Set of quadratic equations? Can be done if finite time, but no efficient algorithm known

Remarks

- Finding **real roots** in many dimensions
  - Set of linear equations? Can be done efficiently (polynomial time)

$$N \leq \text{Polynomial}(\text{dimension},\ \#\ \text{of equations})$$

  - Set of quadratic equations? Can be done if finite time, but no efficient algorithm known

- Finding **integer roots** in many dimensions
  - Set of linear equations? Can be done efficiently (polynomial time)

Remarks

- Finding **real roots** in many dimensions
  - Set of linear equations? Can be done efficiently (polynomial time)

$$N \leq \text{Polynomial}(\text{dimension},\ \#\ \text{of equations})$$

  - Set of quadratic equations? Can be done if finite time, but no efficient algorithm known

- Finding **integer roots** in many dimensions
  - Set of linear equations? Can be done efficiently (polynomial time)
  - Set of quadratic equations? e.g.

$$2x^2 + 2x + 1 = y^2$$

Not possible in finite time!

Remarks

What about solving systems of inequalities?

- is even harder
- But interestingly, the linear case over the reals can still be solved efficiently (this is called linear programming)
- For integer solutions, no efficient algorithm known