CSDS 440: Machine Learning

Soumya Ray (he/him, sray@case.edu)

Olin 516

Office hours T, Th 11:15-11:45 or by appointment

Recap

- Functions taking multiple inputs are called m____ functions. The symbol × indicates the C____ product.
- Functions producing multiple outputs are called v___ functions.
- When a function takes multiple inputs, we compute p_____ derivatives by v____ each input and holding the others f____. A function with m inputs will have (how many) p___ derivatives.
- The r___ v____ containing all p___ derivatives is called the J____ of the function.
- A vector function can be viewed as a v____ of s____ functions.
- The J____ of a vector function is a c____ v____.
- The J____ of a multivariate vector function with n inputs and m outputs is a m___ of size(m/n, m/n).
- The partial derivative of f(g(x,y)) wrt x is ___. This is the partial derivative c__ r___.
- The second derivative is the r___ of c___ of the ____ around a point. Geometrically this refers to the "c____" of a function. If positive the function looks like a ____, if negative it looks like a _____.
- For a function taking m inputs, the Jacobian can be interpreted as a m____ v____ function taking (how many) inputs and producing (how many) outputs.
- For a multivariate function with n inputs, the second derivative is a m____ of size(x,y). It is called the H____ m___.
- If the input is a multivariate vector function, the H___ m___ is a m__-d___ m___. These are called "t____".
- In practice we sometimes convert a t___ into a matrix or vector. This is called r_____.
- In an optimization problem we look for the e____ v____ of a function. Sometimes we look for "argmin" or "argmax" which are the _______.
- Optimization problems can be d____ or c_____. They can also be c____ or u______ .

Review of Calculus and Optimization

- Calculus classes/CSDS 477 / MATH 427/ MATH 433 for the less-crashy version

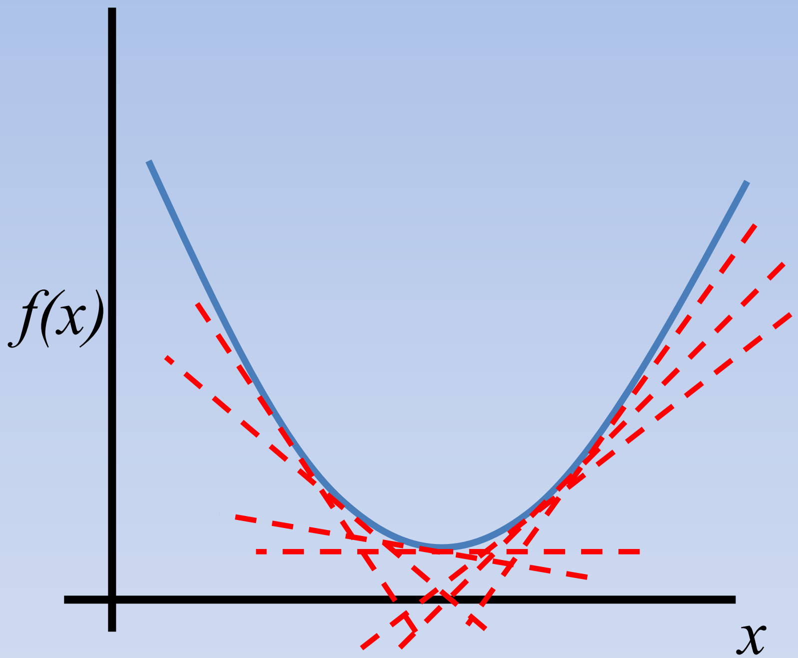
Unconstrained Continuous Optimization

Function of one variable:

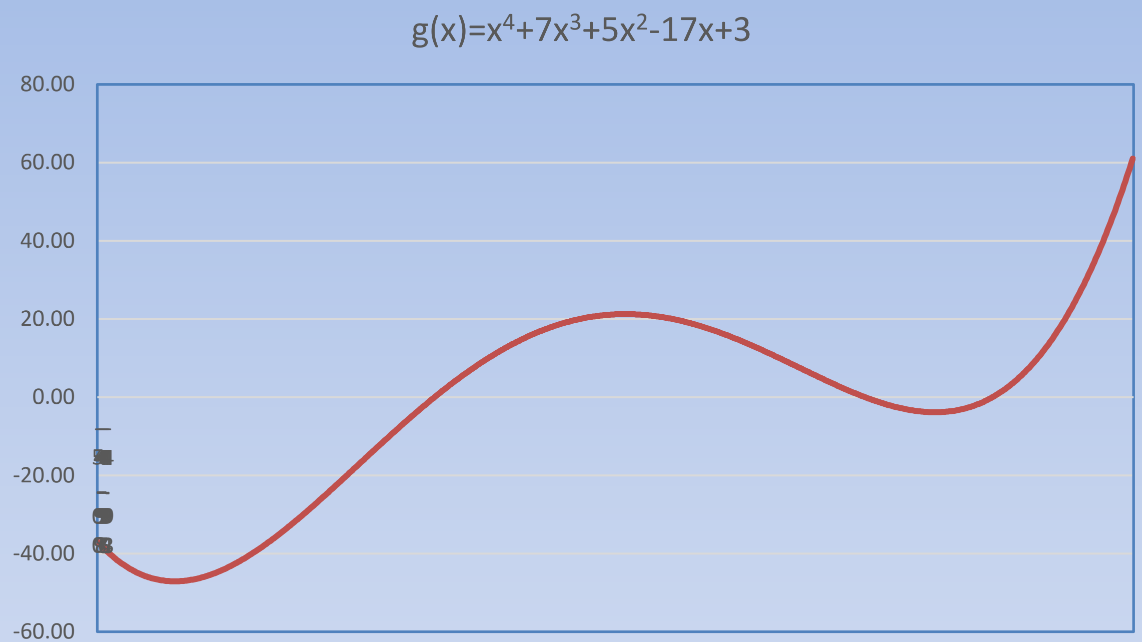
$$\min_x f(x)$$

$$\frac{df}{dx} = 0$$

$$\frac{d^2 f}{dx^2} > 0$$



Example



$$g'(x) = 4x^3 + 21x^2 + 10x - 17$$

$$g'(x) = 0 \text{ for } x = -4.5, -1.4, 0.7$$

$$g''(x) = 12x^2 + 42x + 10$$

$$= 64, -25.28, 45.28$$

Multivariate functions

$$\min_{x_1,\ldots,x_m} f(x_1,\ldots,x_m)$$

$$J = \left(\frac{\partial f}{\partial x_i}\right) = 0$$

Jacobian is zero

$$H = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}\right) > 0$$

Hessian is “positive definite”

Example

$$f(x,y,z) = x^5 y^4 - z^6 y^3 + x^4 z^3$$

$$\nabla f = \begin{bmatrix} 5x^4 y^4 + 4x^3 z^3 & 4x^5 y^3 - 3z^6 y^2 & -6z^5 y^3 + 3x^4 z^2 \end{bmatrix} = 0$$

????

Observation

- In general, analytically solving for the zeros of the Jacobian is computationally (sometimes algebraically!) infeasible

- Alternative: switch to an *iterative* method

Iterative Optimization (One variable)

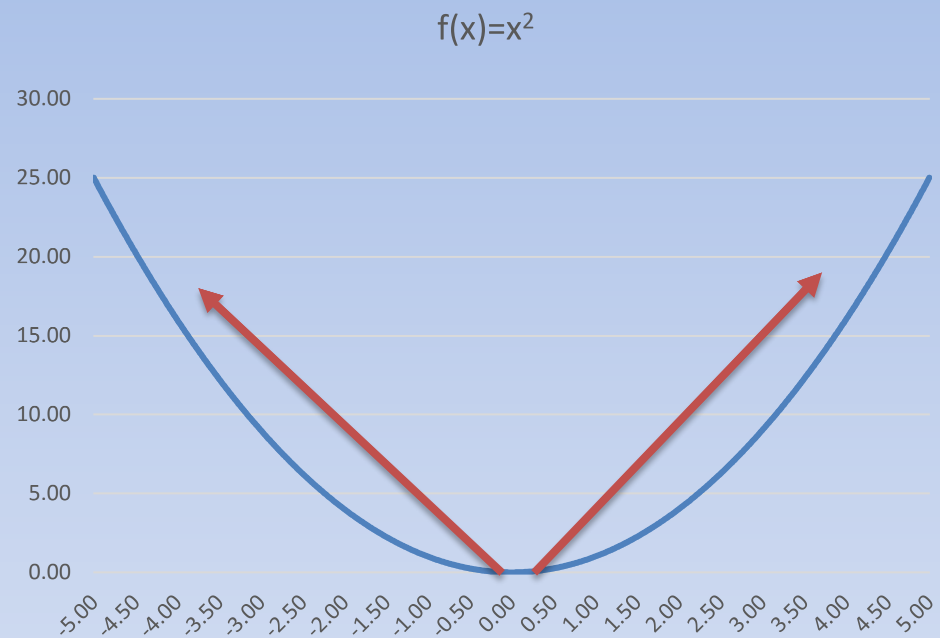
- *Initialize* the solution candidate with a *random guess*
- Until we find the maximum or minimum ("convergence") loop:
  1. Choose a *direction* d
  2. Choose a *stepsize* λ
  3. Move the current guess by λ in the d direction
  4. Check: are we at a minimum/maximum?

Different optimization algorithms will do these steps differently

- By evaluating $\frac{df}{dx} = 0$ at the current guess, and ensuring $\frac{d^2f}{dx^2} \geq 0$ (if minimum) or $\frac{d^2f}{dx^2} \leq 0$ (if maximum)
- In a computer, always check $\left|\frac{df}{dx}\right| \leq tolerance$ (a small quantity such as $1\text{e-}6$)

The derivative as a vector

“take a step in the derivative direction”



- Suppose $x=-1$, $f(x)=1$. Then $f'(x)=-2$. Let $y=x+f'(x)=-3$. $f(y)=9$.
- Suppose $x=1$, $f(x)=1$. $f'(x)=2$. $y=x+f'(x)=3$. $f(y)=9$.

At each point, the derivative can be **viewed as (the slope of) a vector** pointing in the direction of *fastest increase* of the function value

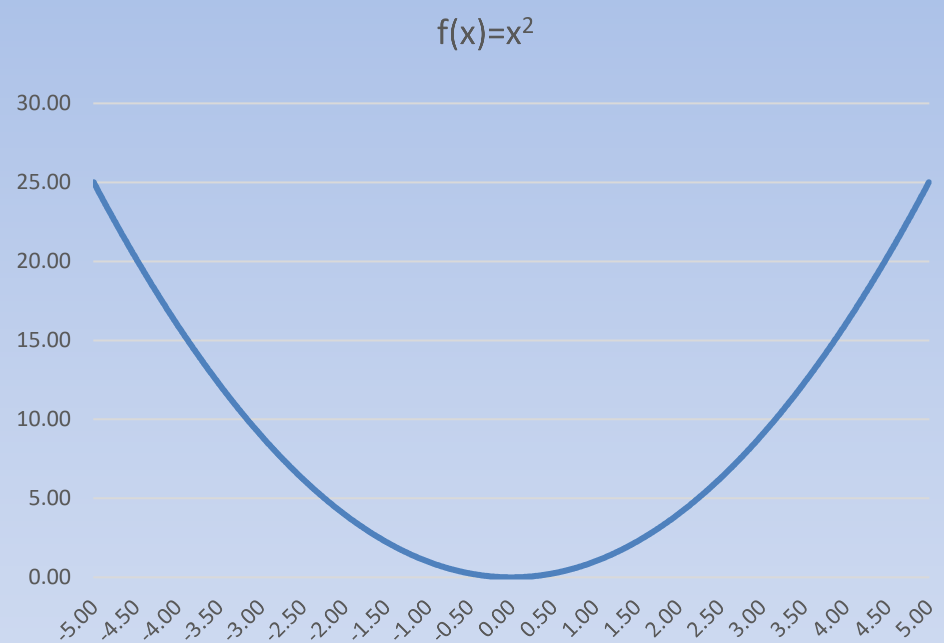
Gradient Ascent/Descent

- From the current x, move in the gradient direction (for maximization) or negative gradient direction (for minimization)

$$x_{new} = x_{old} - \lambda \left. \frac{df}{dx} \right|_{x_{old}}$$

Stepsize

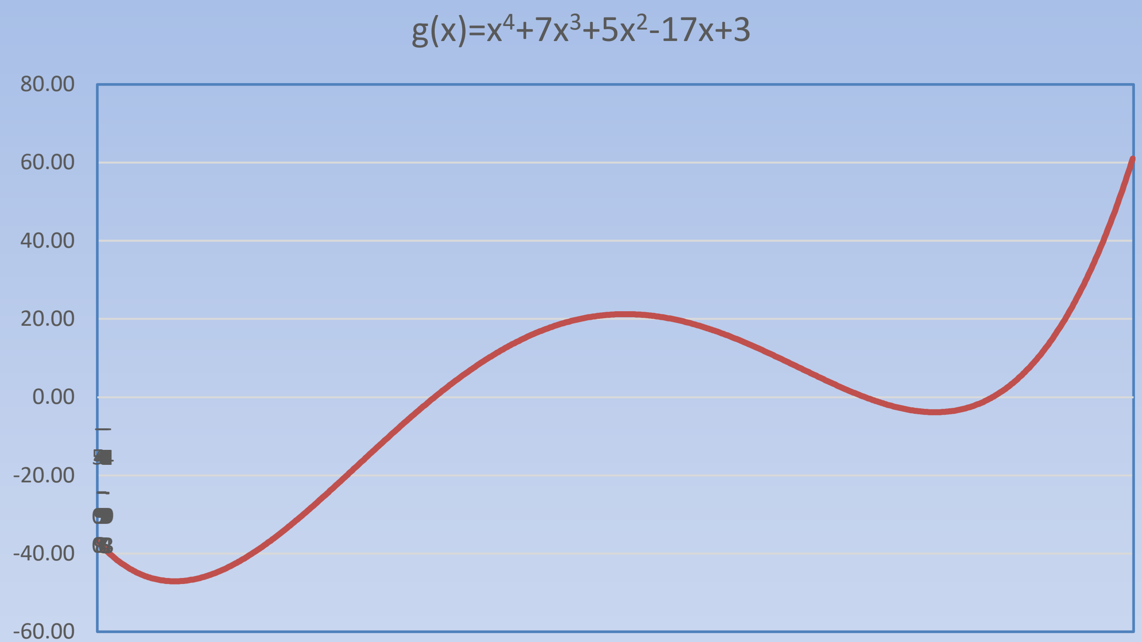
Example



Observations:
1. Stepsize and oscillations
2. Rate of convergence
3. Choice of starting point and solution found

- Set stepsize=0.25, tolerance=0.02
- Start with x_0=−0.4, $f(x)$=0.16, $f'(x)$= −0.8.
- $x_1= x_0-(0.25)(-0.8)= -0.4 +0.2= -0.2$.
- $f(x_1)$=0.04. $f'(x_1)$= −0.4. (not converged)
- $x_2= x_1-(0.25)(-0.4)= -0.2+0.1= -0.1$.
- $f(x_2)$=0.01. $f'(x_2)$= −0.2. (not converged)
- $x_3= x_2-(0.25)(-0.2)= -0.1+0.05= -0.05$.
- $f(x_3)$=0.0025. $f'(x_3)$= −0.1. (not converged)
- $x_4= x_3-(0.25)(-0.1)= -0.05 +0.025= -0.025$.
- $f(x_4)$=6.25e-3. $f'(x_4)$= −0.05. (not converged)
- $x_5= x_4-(0.25)(-0.05)= -0.025+0.0125= -0.0125$.
- $f(x_5)$=1.56e-3. $f'(x_5)$= −0.025. (not converged)
- $x_6= x_5-(0.25)(-0.025)= -0.0125+0.00625= -0.00625$.
- $f(x_6)$=3.9e-4. $f'(x_6)$= −0.015. (converged)

Example



$$g'(x) = 4x^3 + 21x^2 + 10x - 17$$

$$g'(x) = 0 \text{ for } x = -4.5, -1.4, 0.7$$

$$g''(x) = 12x^2 + 42x + 10$$

$$= 64, -25.28, 45.28$$

Newton-Raphson Method

- From the current $\mathbf{x}$, take a Newton step:

$$f(\mathbf{x}_{old}+u)=f(\mathbf{x}_{old})+u^T\nabla f_{\mathbf{x}_{old}}(\mathbf{x})+\frac{1}{2}u^T\nabla^2 f_{\mathbf{x}_{old}}(\mathbf{x})u=g(u)$$

Set $\dfrac{\partial g}{\partial u}=0,$ then

$$\nabla f_{\mathbf{x}_{old}}(\mathbf{x})+\nabla^2 f_{\mathbf{x}_{old}}(\mathbf{x})u=0$$

and

$$u=\underline{-\left[\nabla^2 f_{\mathbf{x}_{old}}(\mathbf{x})\right]^{-1}\nabla f_{\mathbf{x}_{old}}(\mathbf{x})} \longleftarrow \boxed{\text{Newton Step}}$$

$$\mathbf{x}_{new}=\mathbf{x}_{old}-\left[\nabla^2 f_{\mathbf{x}_{old}}(\mathbf{x})\right]^{-1}\nabla f_{\mathbf{x}_{old}}(\mathbf{x})$$

Properties of the NR method

- Fast convergence close to solution

- Not guaranteed to converge if started far from solution, may cycle or diverge in this case

Quasi-Newton methods

- Often, constructing the Hessian for a multivariate function is computationally difficult, because it takes $O(n^2)$ space and time and has to be done over and over

- So a number of methods exist that approximate the Hessian by using the Jacobian at nearby points

Random Restarts

- The solution we get from Gradient Ascent/Descent depends on the initialization

- One way to make it less dependent is to use *random restarts*
  - Run multiple gradient descents with *different, random initializations* and keep the best overall solution
  - Can be done in parallel