Optimization

by Prof. Seungchul Lee
Industrial AI Lab
http://isystems.unist.ac.kr/
POSTECH

Table of Contents

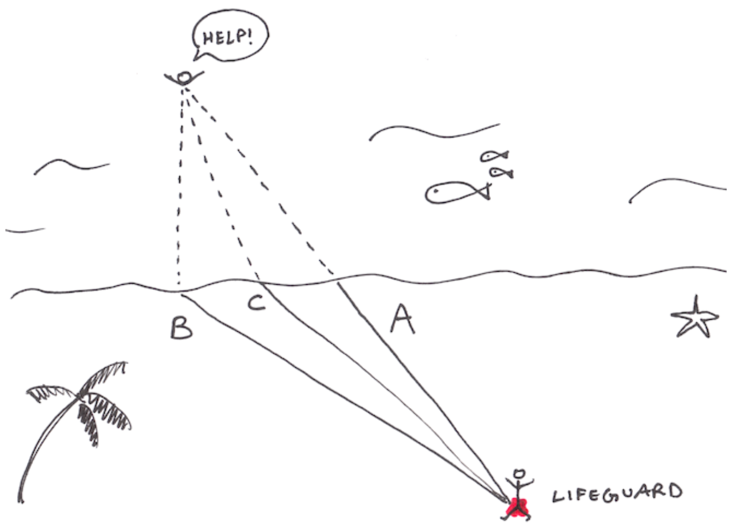
1. Optimization

- an important tool in

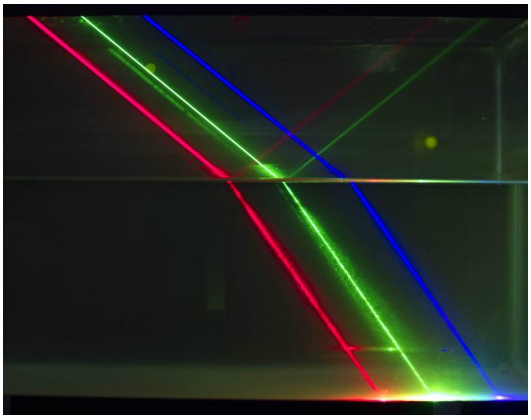
  1) engineering problem solving and

  2) decision science

- People optimize



- Nature optimizes



(source: http://nautil.us/blog/to-save-drowning-people-ask-yourself-what-would-light-do (http://nautil.us/blog/to-save-drowning-people-ask-yourself-what-would-light-do))

3 key components

1. objective
2. decision variable or unknown
3. constraints

Procedures

1. The process of identifying objective, variables, and constraints for a given problem is known as "modeling"
2. Once the model has been formulated, optimization algorithm can be used to find its solutions.

In mathematical expression

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \qquad i = 1, \cdots, m \end{aligned}$$

- $x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$ is the decision variable
- $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is objective function
- Feasible region : $C = \{x : g_i(x) \leq 0, i = 1, \cdots, m\}$
- $x^* \in \mathbb{R}^2$ is an optimal solution if $x^* \in C$ and $f(x^*) \leq f(x), \forall x \in C$

Remarks : equivalent

$$\min_x f(x) \quad \leftrightarrow \quad \max_x -f(x)$$

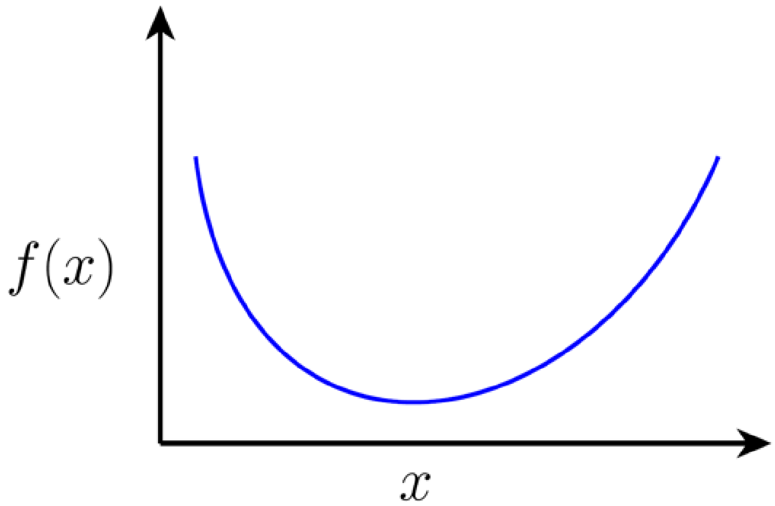
$$g_i(x) \leq 0 \quad \leftrightarrow \quad -g_i(x) \geq 0$$

$$h(x) = 0 \quad \leftrightarrow \quad \begin{cases} h(x) \leq 0 & \text{and} \\ h(x) \geq 0 \end{cases}$$

The good news: for many classes of optimization problems, people have already done all the "hardwork" of developing numerical algorithms

2. Solving Optimization Problems

- Starting with th unconstrained, one dimensional case



  - To find minimum point x^*, we can look at the derivave of the function $f'(x)$: any location where $f'(x)$ = 0 will be a "flat" point in the function

- For convex problems, this is guaranteed to be a minimum

- Generalization for multivariate function $f : \mathbb{R}^n \rightarrow \mathbb{R}$
  - the gradient of f must be zero

$$\nabla_x f(x) = 0$$

- For defined as above, gradient is a n-dimensional vector containing partial derivatives with respect to each dimension

$$\nabla_x f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}$$

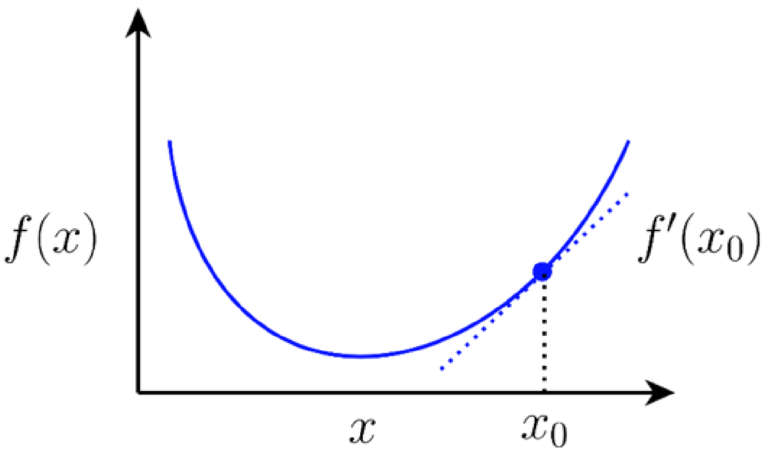
- For continuously differentiable f and unconstrained optimization, optimal point must have $\nabla_x f(x^*) = 0$

3. How do we Find $\nabla_x f(x) = 0$

- Direct solution
  - In some cases, it is possible to analytically compute x^* such that $\nabla_x f(x^*) = 0$

$$\begin{aligned} f(x) &= 2x_1^2 + x_2^2 + x_1x_2 - 6x_1 - 5x_2 \\ \implies \nabla_x f(x) &= \begin{bmatrix} 4x_1 + x_2 + 6 \\ 2x_2 + x_1 + 5 \end{bmatrix} \\ \implies x^\star &= \begin{bmatrix} 4 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 6 \\ 5 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \end{aligned}$$

- Iterative methods
  - More commonly the condition that the gradient equal zero will not have an analytical solution, require iterative methods

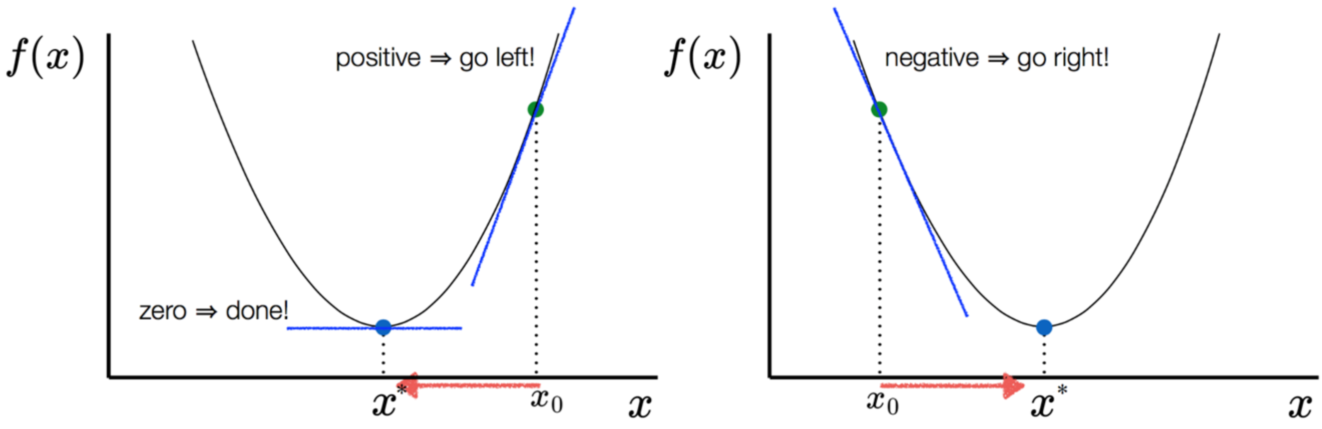


- The gradient points in the direction of "steepest ascent" for function f

4. Gradient Descent

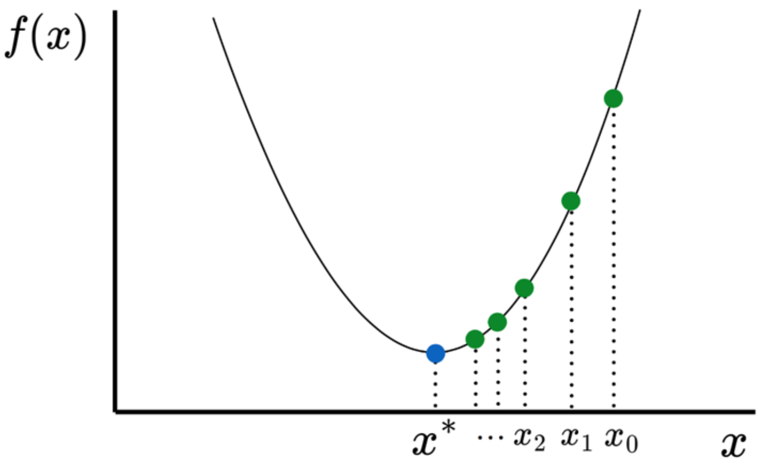
- Decent Direction (1D)
- It motivates the gradient descent algorithm, which repeatedly takes steps in the direction of the negative gradient

$$x \leftarrow x - \alpha \nabla_x f(x) \qquad \text{for some step size } \alpha > 0$$



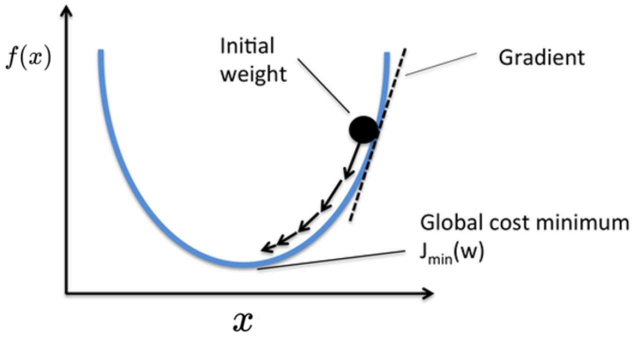
- Gradient Descent

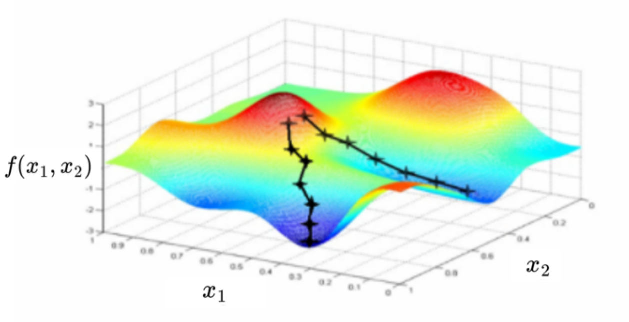
$$\text{Repeat} : x \leftarrow x - \alpha \nabla_x f(x) \qquad \text{for some step size } \alpha > 0$$



- Gradient Descent in High Dimension

$$\text{Repeat} : x \leftarrow x - \alpha \nabla_x f(x)$$





- Gradient Descent Example

$$\begin{aligned} &\min \quad (x_1 - 3)^2 + (x_2 - 3)^2 \\ = &\min \quad \frac{1}{2} \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 6 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 18 \end{aligned}$$

- Update rule

$$X_{i+1} = X_i - \alpha_i \nabla f(X_i)$$

In [1]:

```
import numpy as np
```

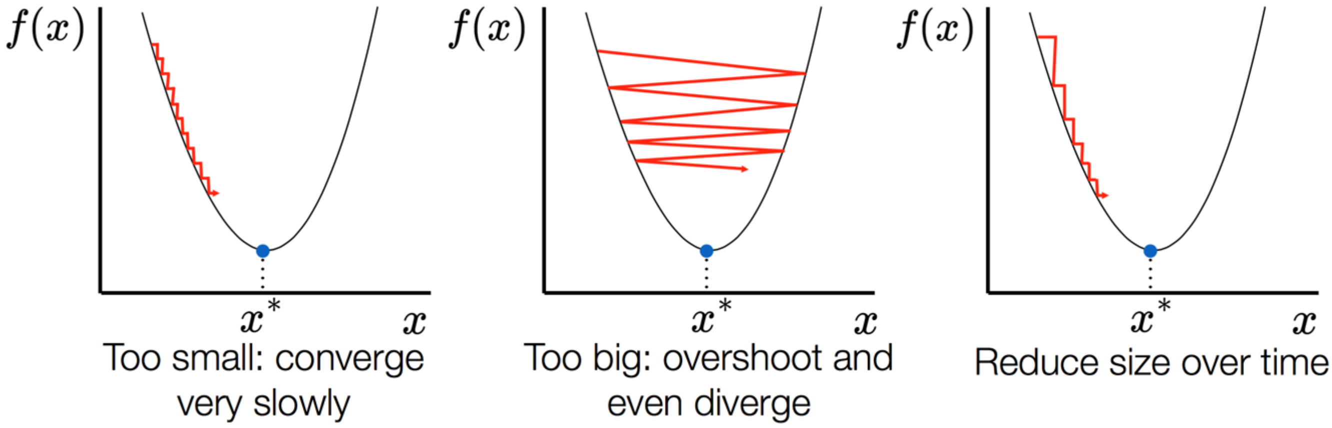
In [2]:

```
H = np.array([[2, 0],[0, 2]])
f = -np.array([[6],[6]])

x =  np.zeros((2,1))
alpha = 0.2

for i in range(25):
    g = H.dot(x) + f
    x = x - alpha*g

print(x)
```

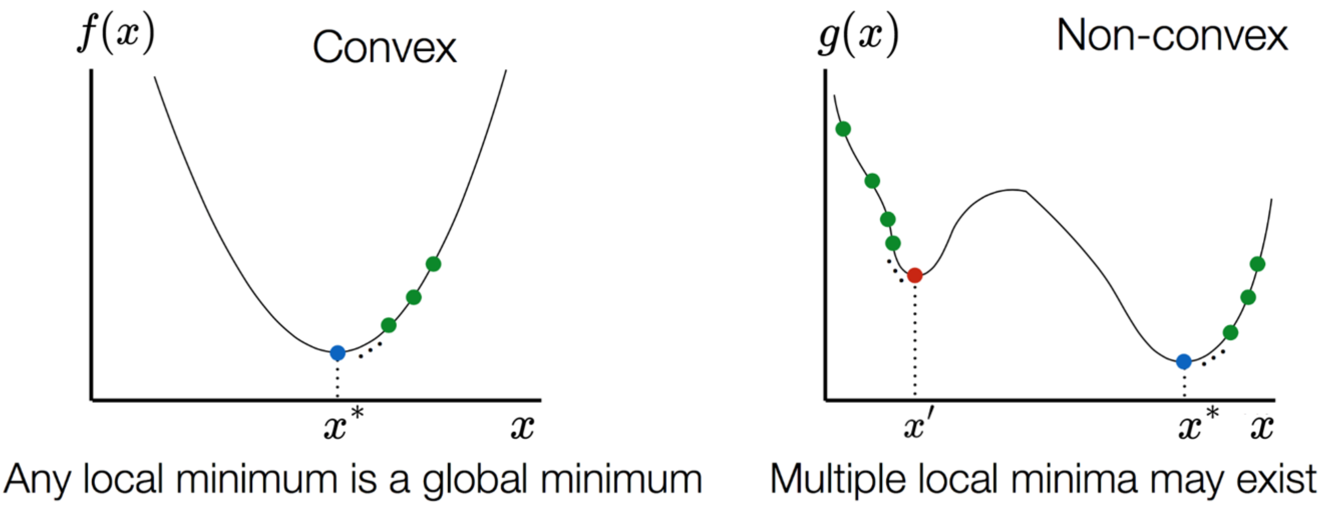
```
[[ 2.99999147]
 [ 2.99999147]]
```

Choosing Step Size α

- Learning rate



Where will We Converge?



5. Practically Solving Optimization Problems

- The good news: for many classes of optimization problems, people have already done all the “hard work” of developing numerical algorithms
  - A wide range of tools that can take optimization problems in “natural” forms and compute a solution

- We will use CVX (or CVXPY) as an optimization solver
  - Only for convex problems
  - Download: [http://cvxr.com/cvx/ (http://cvxr.com/cvx/)](http://cvxr.com/cvx/)

- Gradient descent
  - Neural networks/deep learning

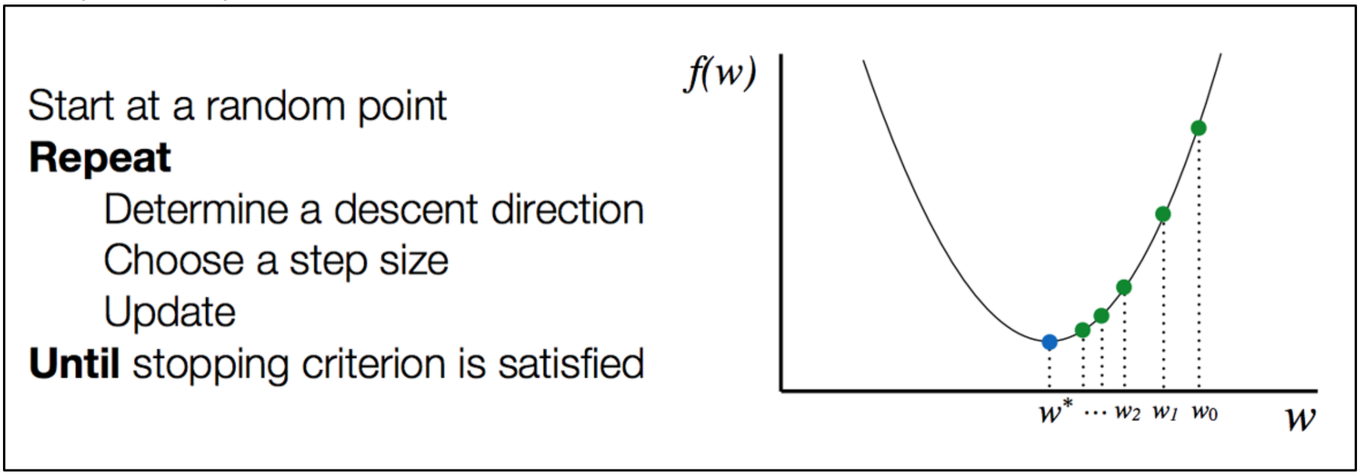
5.1. Gradient Descent vs. Analytical Solution

- Analytical solution for MSE
- Gradient descent
    - Easy to implement
    - Very general, can be applied to any differentiable loss functions
    - Requires less memory and computations (for stochastic methods)

- Gradient descent provides a general learning framework
- Can be used both for classification and regression
- Training Neural Networks: Gradient Descent

5.2. Training Neural Networks

- Optimization procedure



- It is not easy to numerically compute gradients in network in general.
    - The good news: people have already done all the "hard work" of developing numerical solvers (or libraries)
    - There are a wide range of tools
        - We will use TensorFlow

In [3]:

```
%%javascript
$.getScript('https://kmahelona.github.io/ipython_notebook_goodies/ipython_notebook_toc.
js')
```