

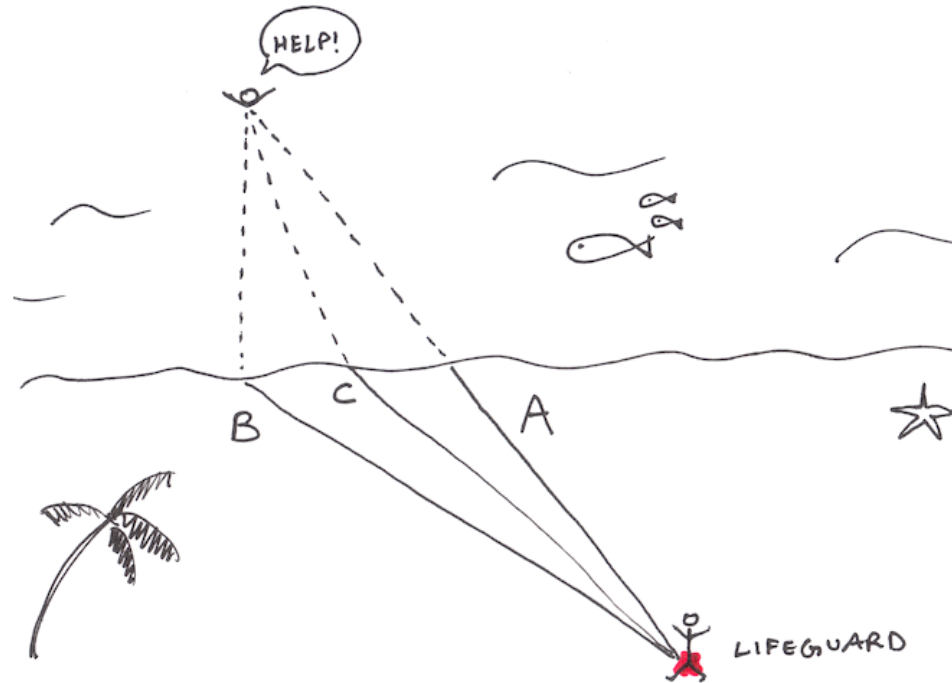


Optimization for Deep Learning

Industrial AI Lab.
Prof. Seungchul Lee

Optimization

- An important tool in
 - 1) Engineering problem solving and
 - 2) Decision science



Optimization

- 3 key components
 - 1) Objective function
 - 2) Decision variable or unknown
 - 3) Constraints
- Procedures
 - 1) The process of identifying objective, variables, and constraints for a given problem (known as "modeling")
 - 2) Once the model has been formulated, optimization algorithm can be used to find its solutions

Optimization: Mathematical Model

- In mathematical expression

$$\begin{array}{ll} \min_x & f(x) \\ \text{subject to} & g_i(x) \leq 0, \quad i = 1, \dots, m \end{array}$$

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

Optimization: Mathematical Model

- In mathematical expression

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

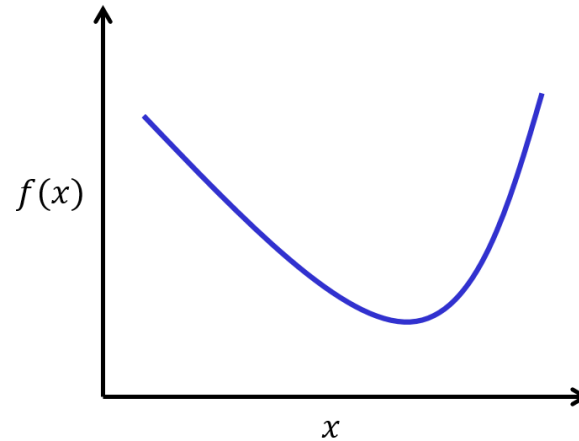
- Remarks: equivalent

$$\begin{aligned} \min_x f(x) & \leftrightarrow \max_x -f(x) \\ g_i(x) \leq 0 & \leftrightarrow -g_i(x) \geq 0 \\ h(x) = 0 & \leftrightarrow \begin{cases} h(x) \leq 0 \\ h(x) \geq 0 \end{cases} \text{ and} \end{aligned}$$

Solving Optimization Problems

Solving Optimization Problems

- Starting with the unconstrained, one dimensional case



- To find minimum point x^* , we can look at the derivative 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

Solving Optimization Problems

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

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

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

- 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

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

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

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

$$f(x) = 2x_1^2 + x_2^2 + x_1 x_2 - 6x_1 - 5x_2$$

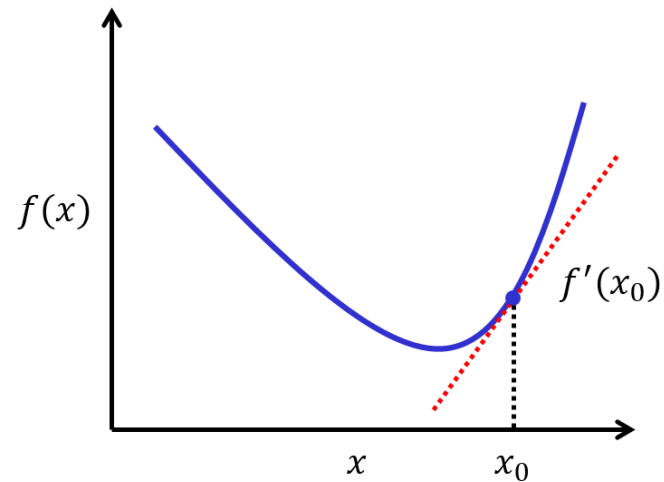
$$\implies \nabla_x f(x) = \begin{bmatrix} 4x_1 + x_2 - 6 \\ 2x_2 + x_1 - 5 \end{bmatrix}$$

$$\implies x^* = \begin{bmatrix} 4 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 6 \\ 5 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

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

- Iterative methods

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

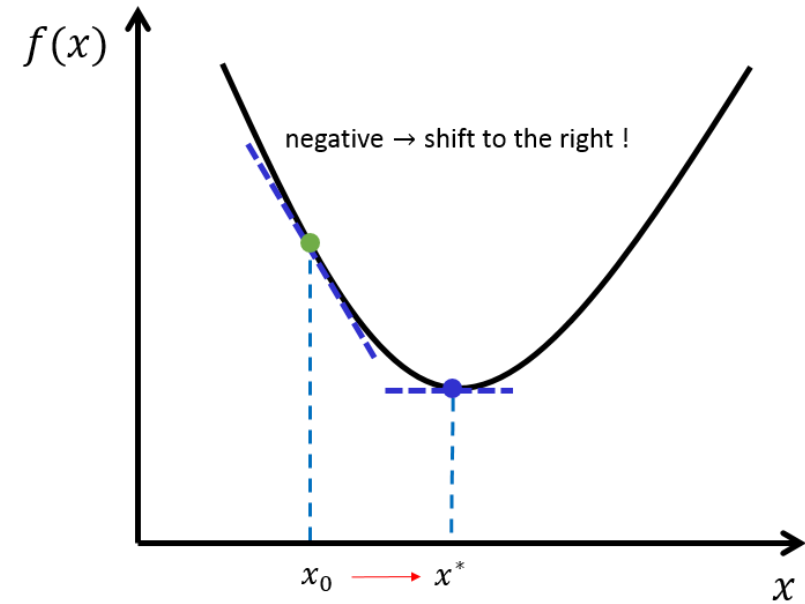
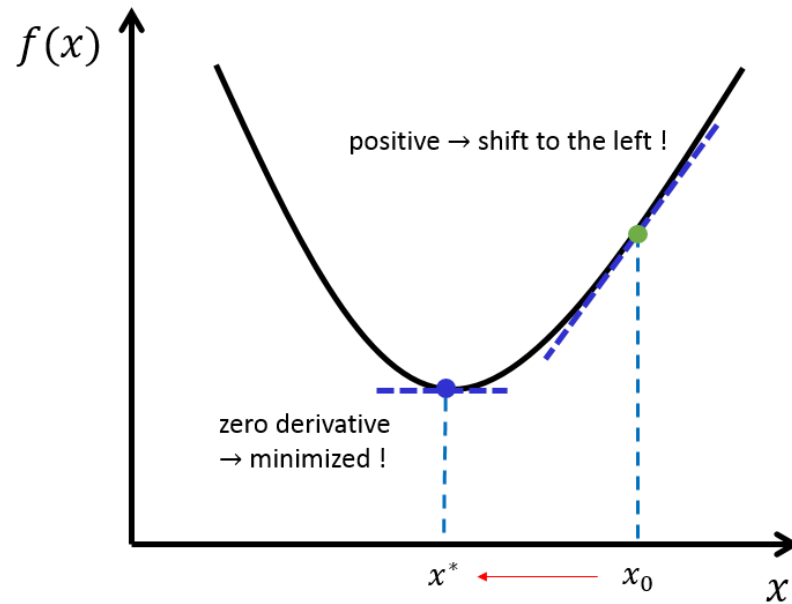


- The gradient points in the direction of “steepest ascent” for function f

Descent Direction (1D)

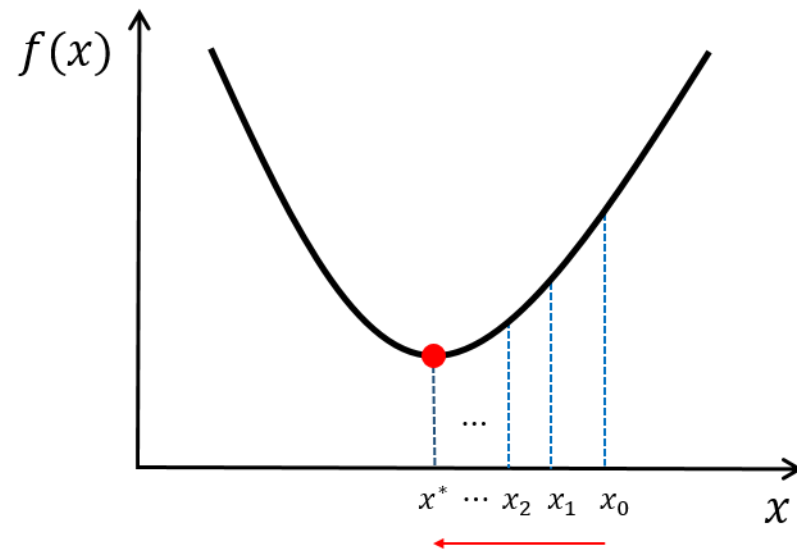
- It motivates the *gradient descent* algorithm, which repeatedly takes steps in the direction of the negative gradient

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



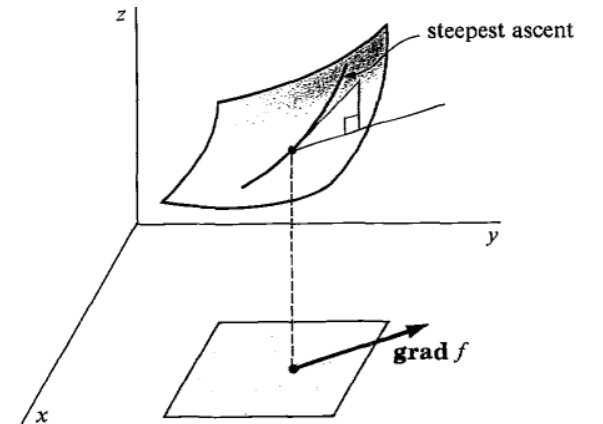
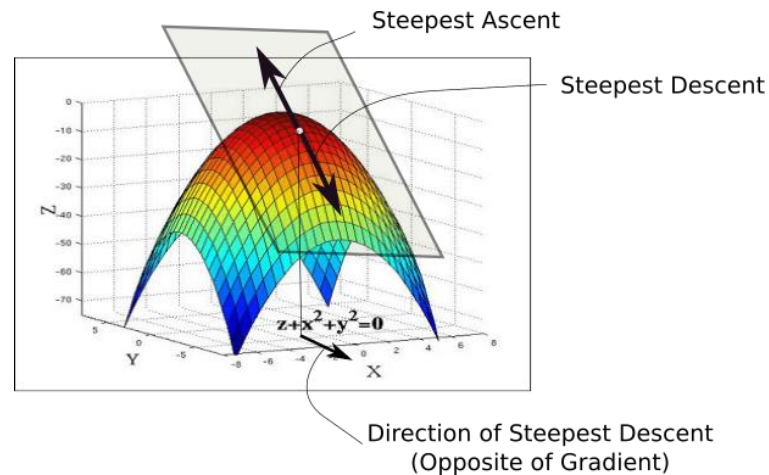
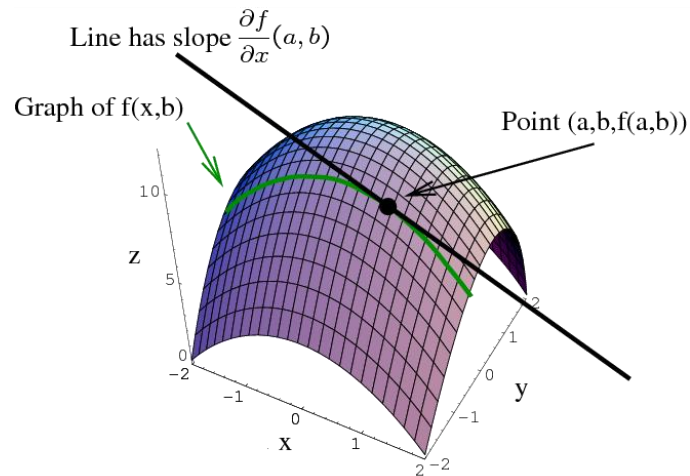
Gradient Descent

Repeat: $x \leftarrow x - \alpha \nabla_x f(x)$ for some *step size* $\alpha > 0$



Gradient Descent in Higher Dimension

Repeat: $x \leftarrow x - \alpha \nabla_x f(x)$ for some *step size* $\alpha > 0$



Gradient Descent

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

$$\begin{aligned} f &= \frac{1}{2} X^T H X + g^T X \\ \nabla f &= H X + g \end{aligned}$$

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

```
H = np.matrix([[2, 0],[0, 2]])
g = -np.matrix([[6],[6]])

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

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

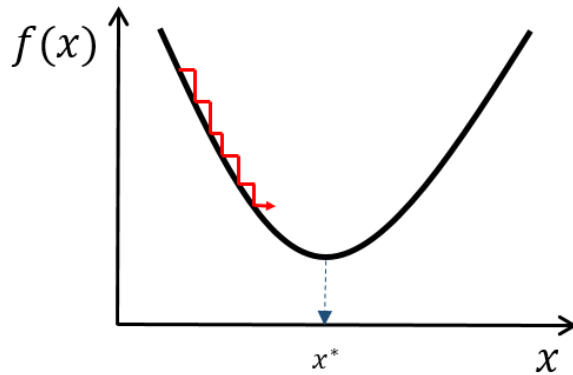
print(x)
```

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

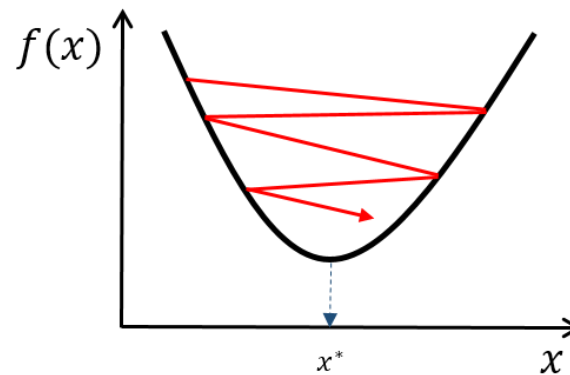
y	$\frac{\partial y}{\partial x}$
Ax	A^T
$x^T A$	A
$x^T x$	$2x$
$x^T Ax$	$Ax + A^T x$

Choosing Step Size α

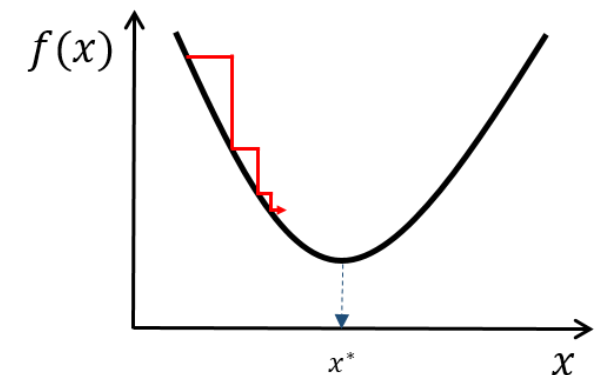
- Learning rate



Too small: converge
very slowly

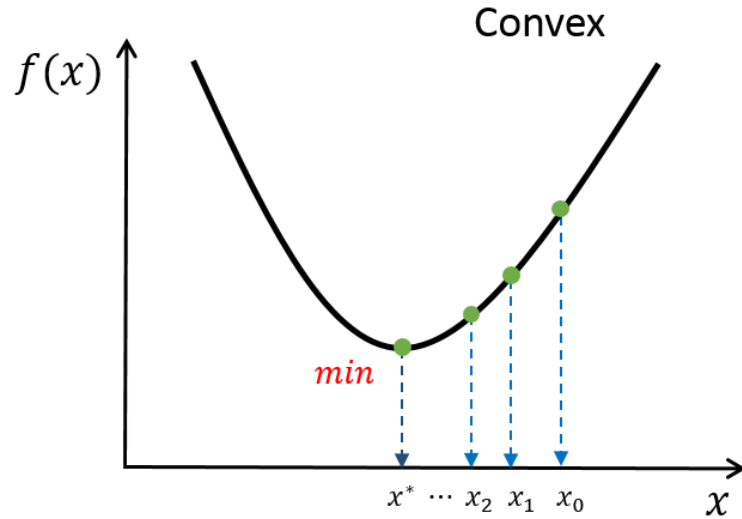


Too big: overshoot and
even diverge

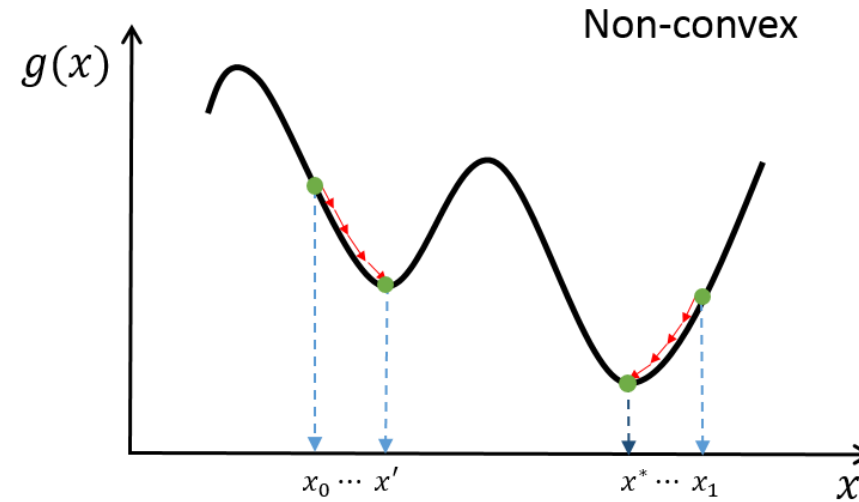


Reduce size over time

Where will We Converge?



Any local minimum is a global minimum



Multiple local minima may exist

- Random initialization
- Multiple trials

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
- Gradient descent
 - Easy to implement
 - Very general, can be applied to any differentiable loss functions
 - Requires less memory and computations (for stochastic methods)
 - Neural networks/deep learning
 - TensorFlow