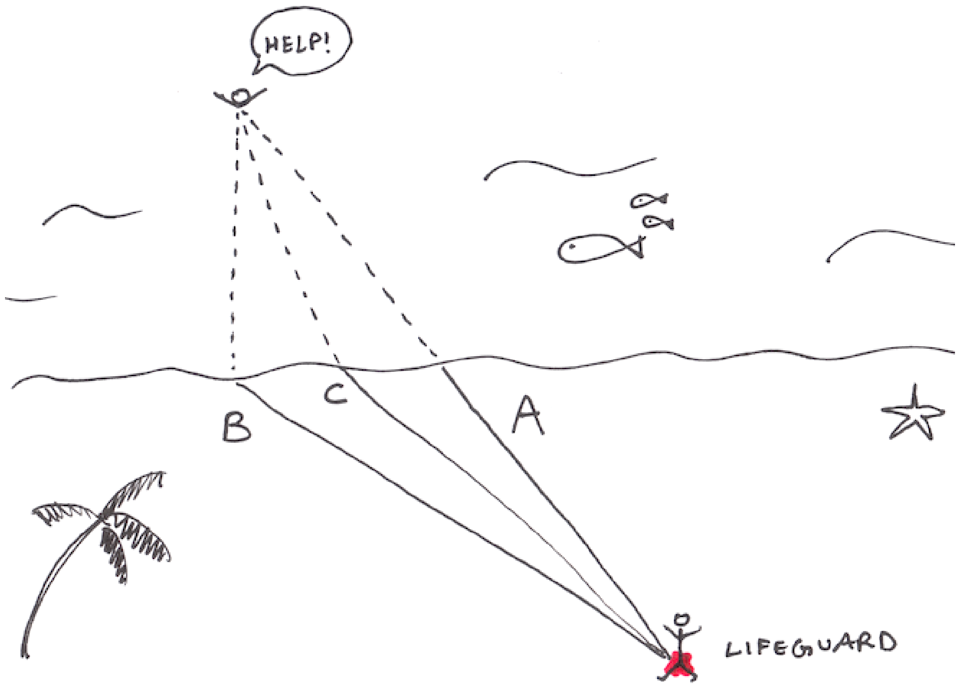
Optimization for Deep Learning

Industrial AI Lab.

Prof. Seungchul Lee

Optimization

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



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

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

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

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

Optimization: Mathematical Model

- In mathematical expression

$$\begin{array}{lll} \min\limits_{x} & f(x) & \\ \text{subject to} & g_i(x) \leq 0, & i = 1, \cdots, m \end{array}$$

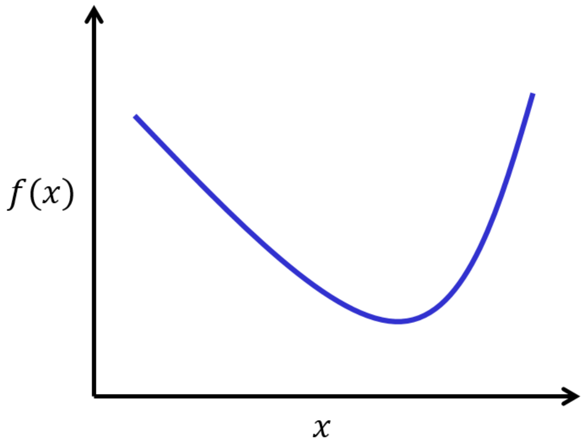
- Remarks: equivalent

$$\begin{array}{rcl} \min\limits_{x} f(x) & \leftrightarrow & \\ g_i(x) \leq 0 & \leftrightarrow & \\ h(x) = 0 & \leftrightarrow & \end{array}$$

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

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

- Generalization for multivariate function $f: \mathbb{R}^n \to \mathbb{R}$
  - 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

- For continuously differentiable f and unconstrained optimization, optimal point must have

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

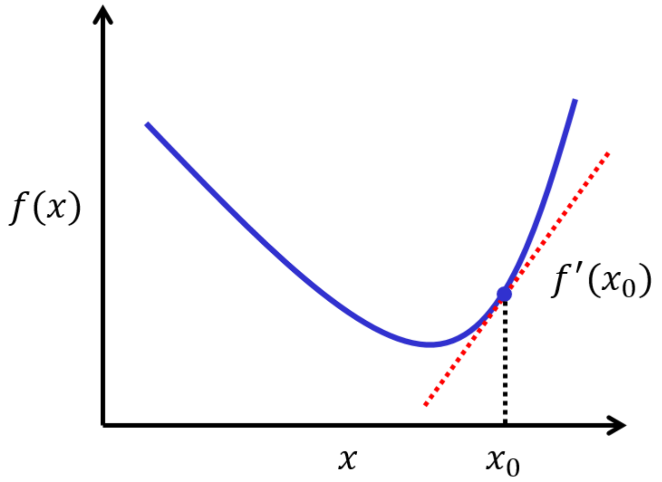
How do we Find $\nabla_x f(x) = 0$

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

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

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

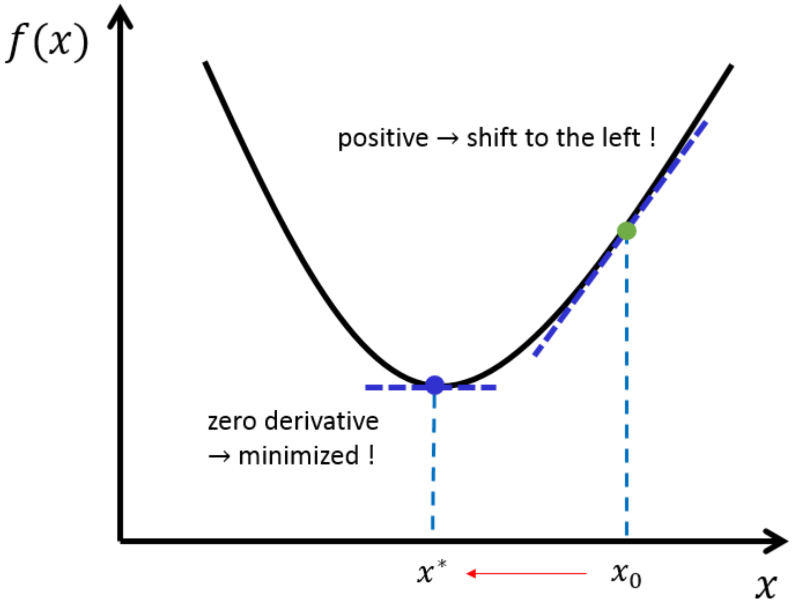


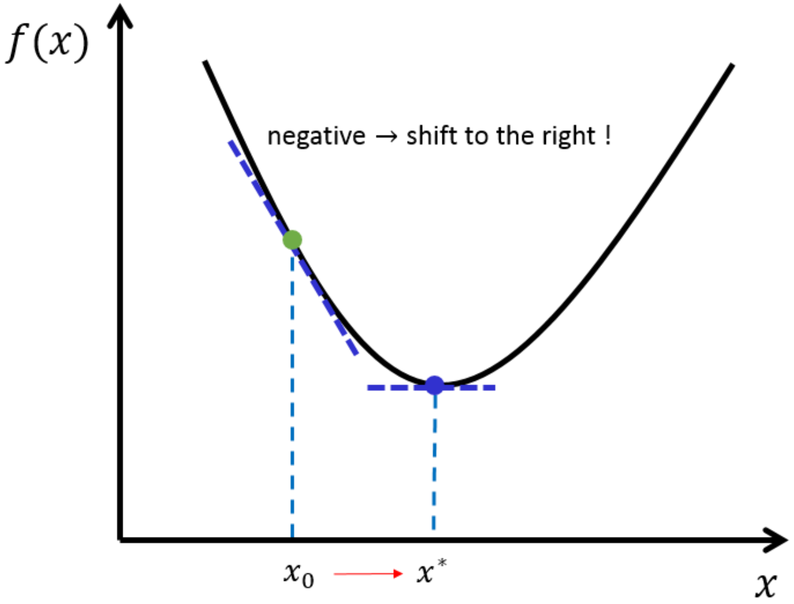
  - The gradient points in the direction of “steepest ascent” for function f

Descent Direction (1D)

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

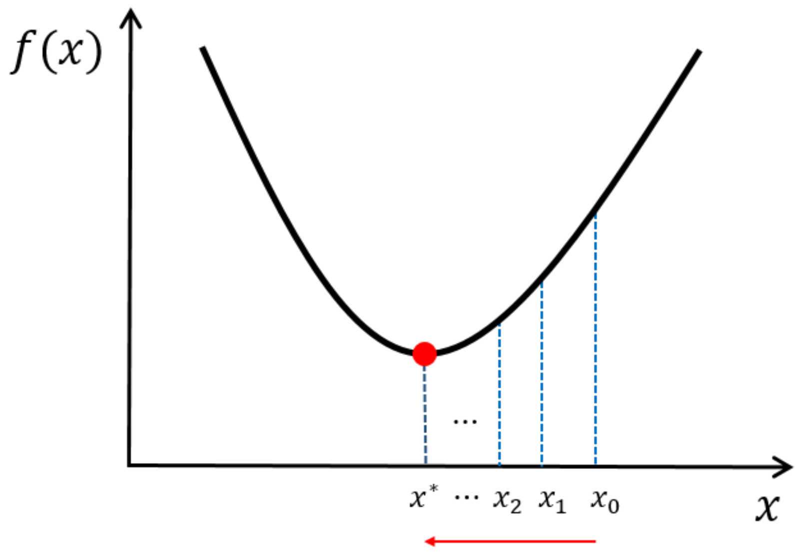
for some *step size* $\alpha > 0$





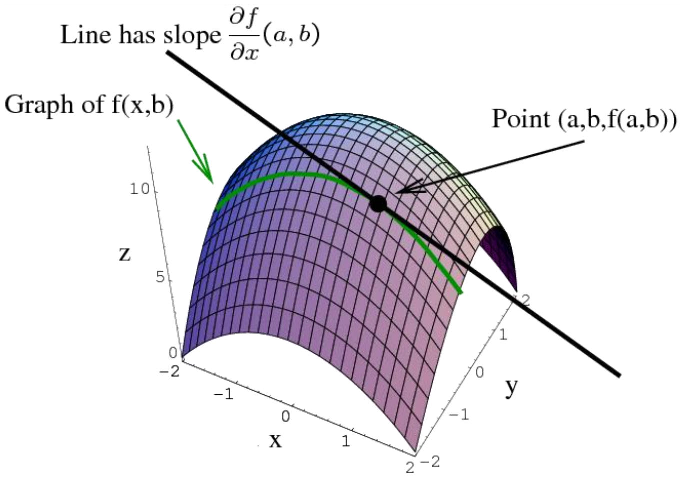
Gradient Descent

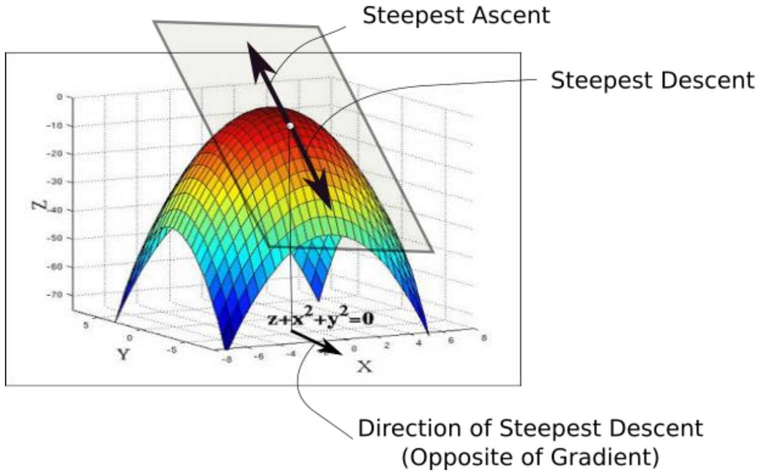
Repeat: $x \leftarrow x - \alpha \nabla_x f(x)$ for some *step size* $\alpha > 0$

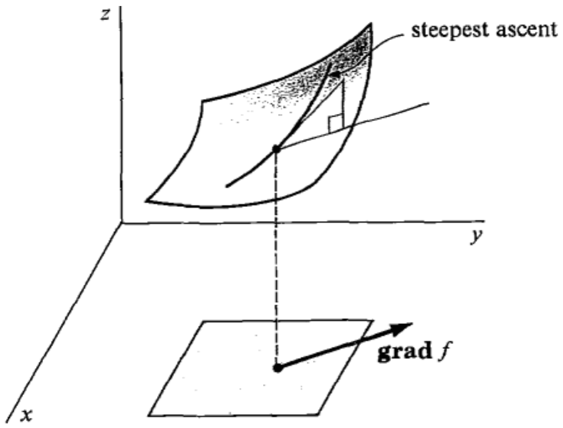


Gradient Descent in Higher Dimension

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







Gradient Descent

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

$$f = \frac{1}{2}X^T H X + g^T X$$
$$\nabla f = HX + g$$

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

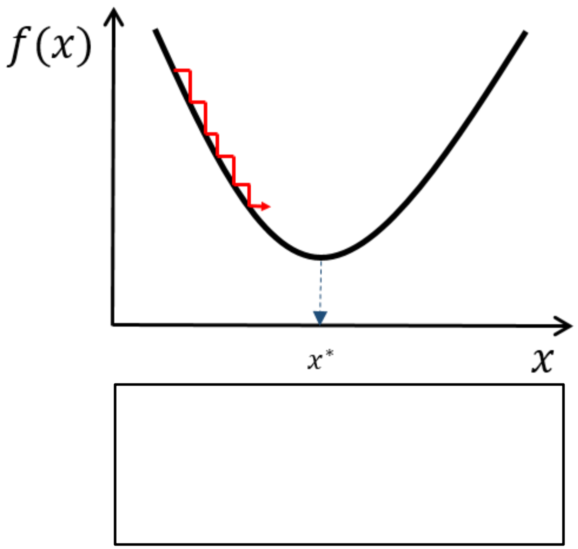
```
H = np.matrix([[2, 0],[0, 2]])
g = -np.matrix([[6],[6]])

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

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

print(x)
```

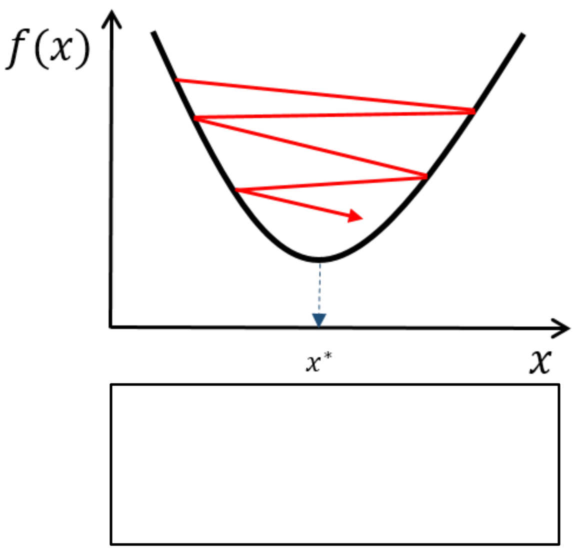
```
[[ 2.99999147]
 [ 2.99999147]]
```

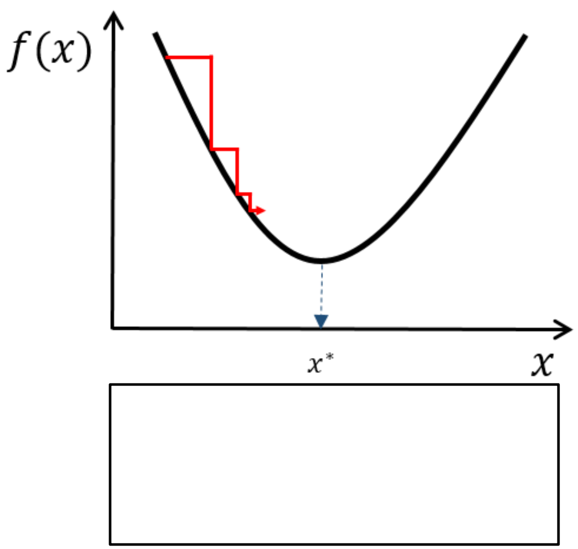
y	$\frac{\partial y}{\partial x}$
Ax	A^T
$x^T A$	A
$x^T x$	$2x$
$x^T A x$	$Ax + A^T x$

Choosing Step Size α

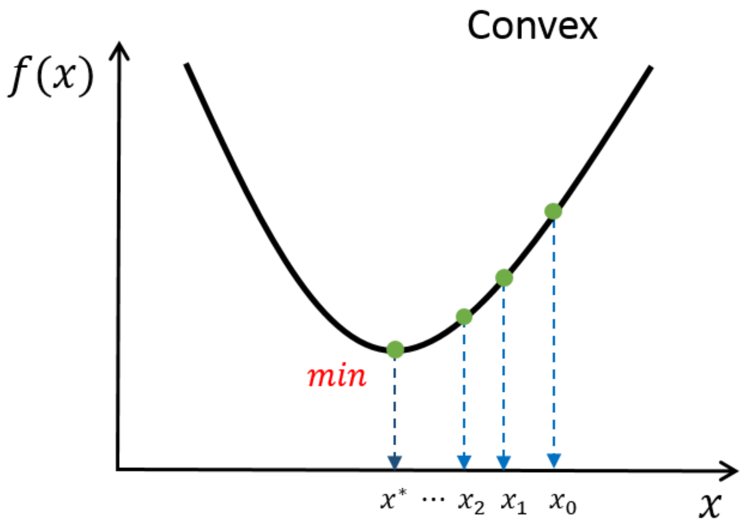
- Learning rate



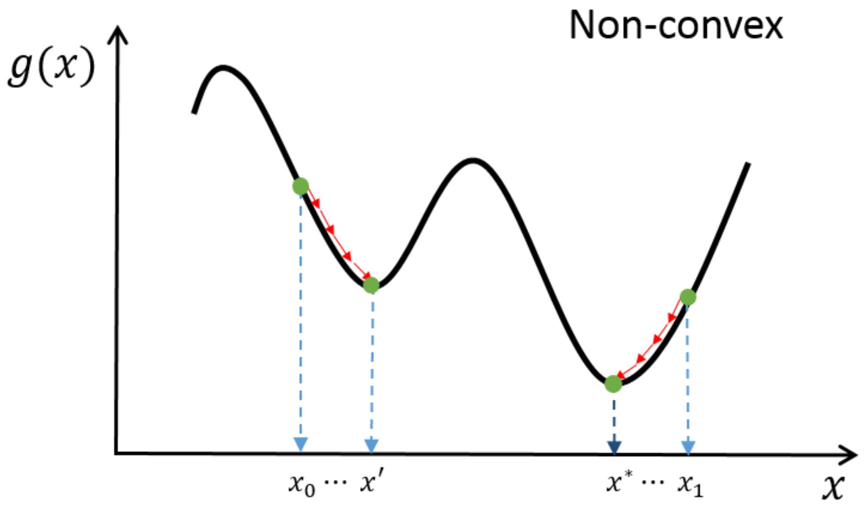




Where will We Converge?



Any local minimum is a global minimum



Multiple local minima may exist

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