Gradient Descent

Industrial AI Lab.
Prof. Seungchul Lee
Yunseob Hwang, Juwon Na

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

Analytic Approach

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

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

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

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

Gradients

- Matrix derivatives

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

How to Find $\nabla_x f(x) = 0$

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

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

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

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

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

Examples

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

- affine function $g(x) = a^T x + b$

$$\nabla g(x) = a, \qquad \nabla^2 g(x) = 0$$

- quadratic function $g(x) = x^T P x + q^T x + r$, $\qquad P = P^T$

$$\nabla g(x) = 2Px + q, \qquad \nabla^2 g(x) = 2P$$

- $g(x) = \|Ax - b\|^2 = x^T A^T A x - 2b^T A x + b^T b$

$$\nabla g(x) = 2A^T A x - 2A^T b, \qquad \nabla^2 g(x) = 2A^T A$$

Revisit: Least-Square Solution

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

- Scalar Objective: $J = \|Ax - y\|^2$

$$\begin{aligned} J(x) &= (Ax - y)^T (Ax - y) \\ &= \left(x^T A^T - y^T\right)(Ax - y) \\ &= x^T A^T A x - x^T A^T y - y^T A x + y^T y \end{aligned}$$

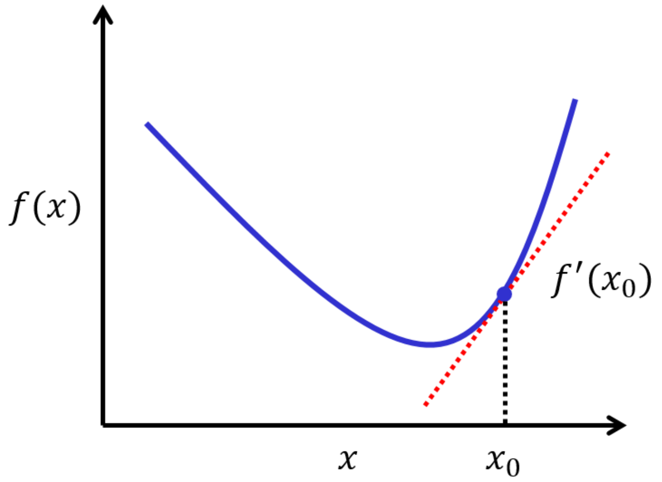
$$\begin{aligned} \frac{\partial J}{\partial x} &= A^T A x + \left(A^T A\right)^T x - A^T y - \left(y^T A\right)^T \\ &= 2A^T A x - 2A^T y = 0 \end{aligned}$$

$$\Longrightarrow \quad \left(A^T A\right) x = A^T y$$

$$\therefore \quad x^* = \left(A^T A\right)^{-1} A^T y$$

Iterative Approach

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



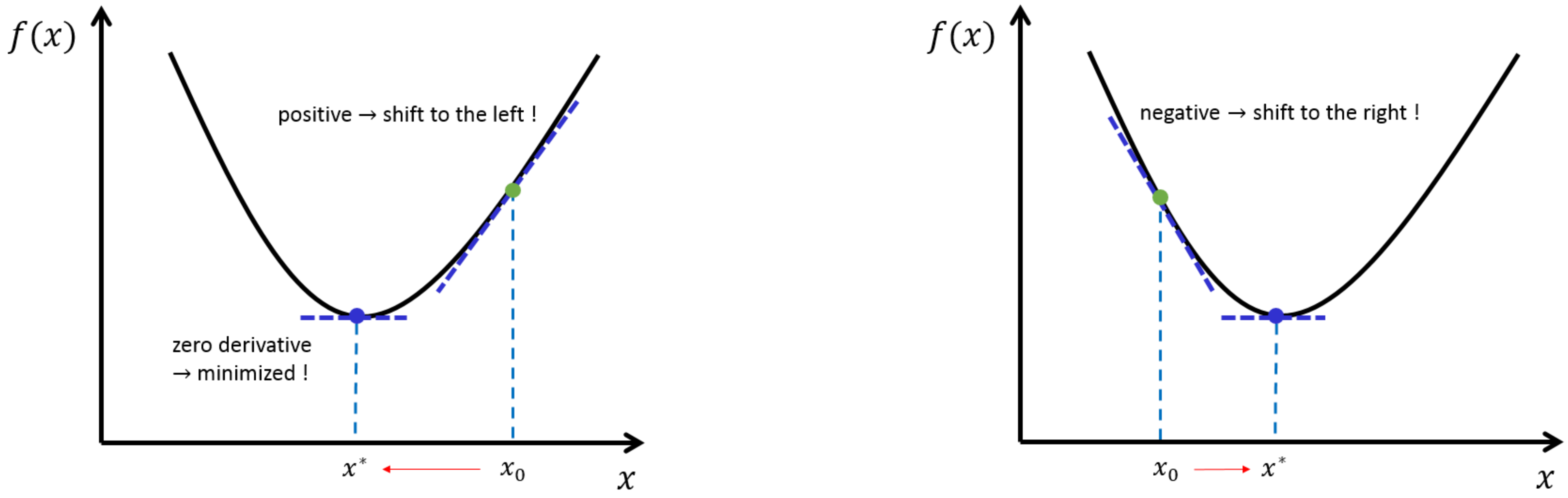
  - The gradient points in the direction of “steepest ascent” for function f

Gradient Descent

Descent Direction (1D)

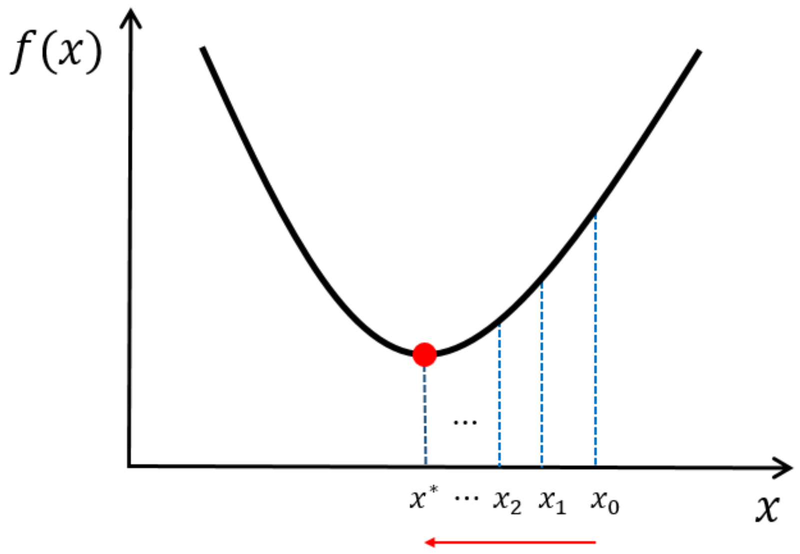
- It motivates the *gradient descent* algorithm, which repeatedly takes steps in the direction of the negative gradient

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



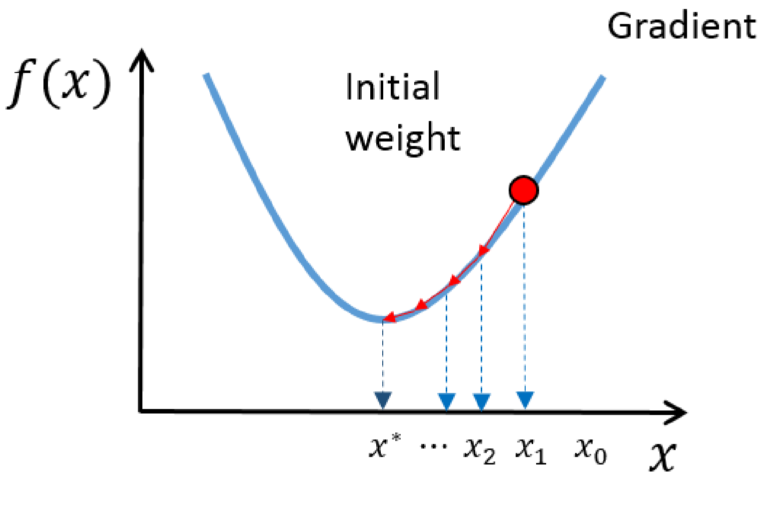
Gradient Descent

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

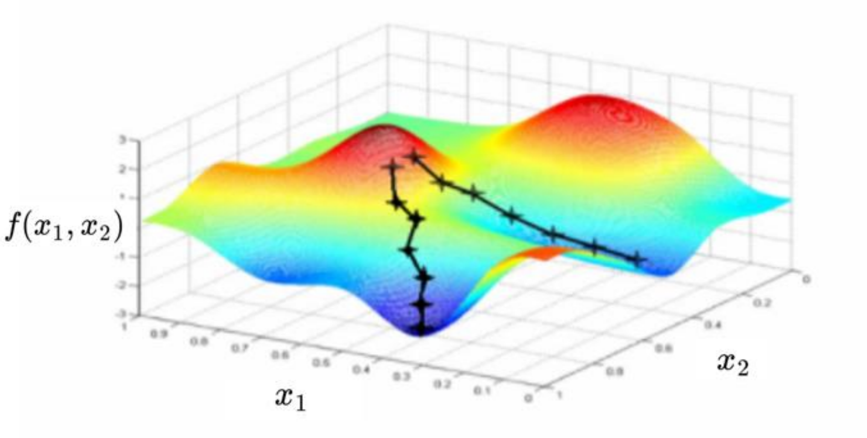


Gradient Descent in High Dimension

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

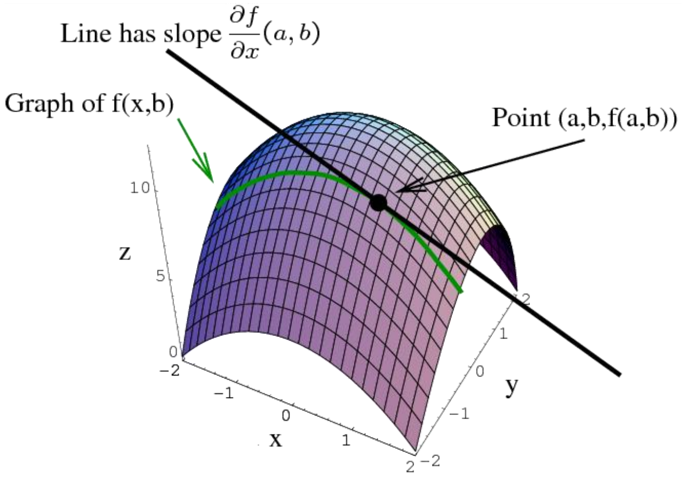


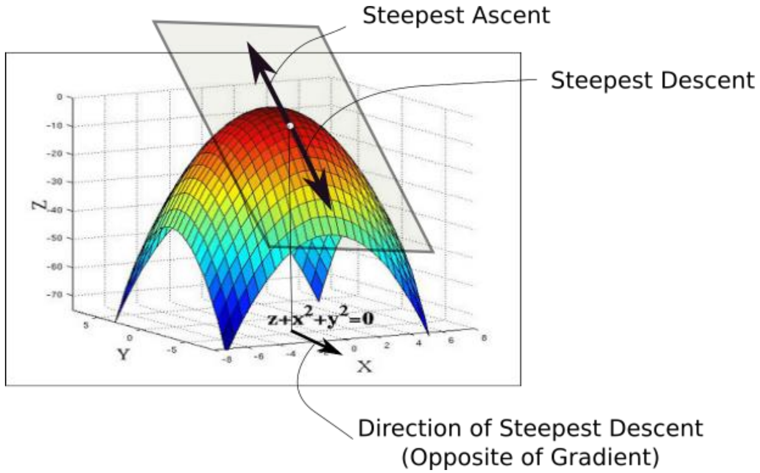
Global cost minimum $J_{\min}(\omega)$

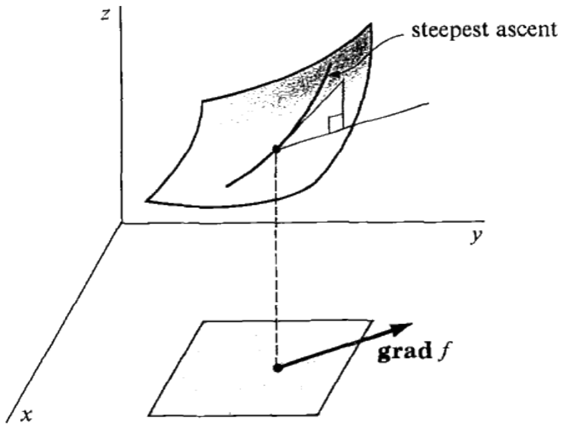


Gradient Descent in High Dimension

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







Gradient Descent

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

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

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

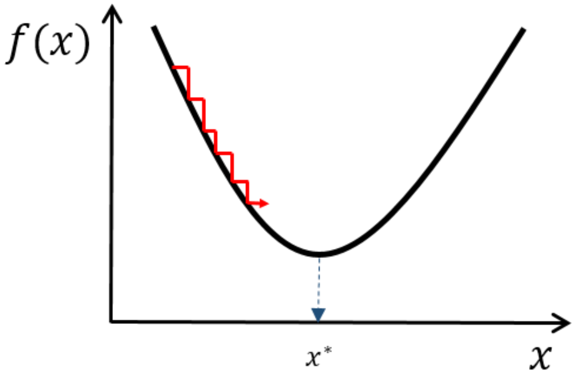
```
H = np.matrix([[2, 0],[0, 2]])
g = -np.matrix([[6],[6]])

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]]
```

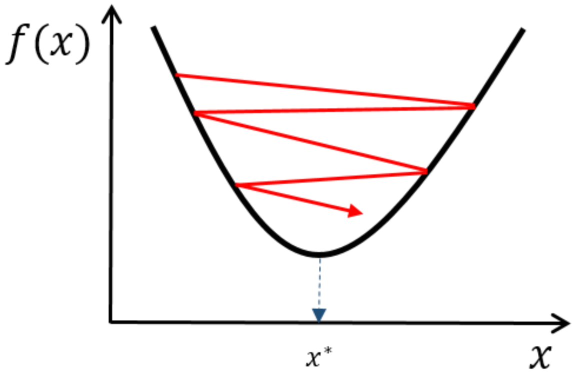
y	$\frac{\partial y}{\partial x}$
Ax	A^T
$x^T A$	A
$x^T x$	$2x$
$x^T A x$	$Ax + A^T x$

Choosing Step Size α

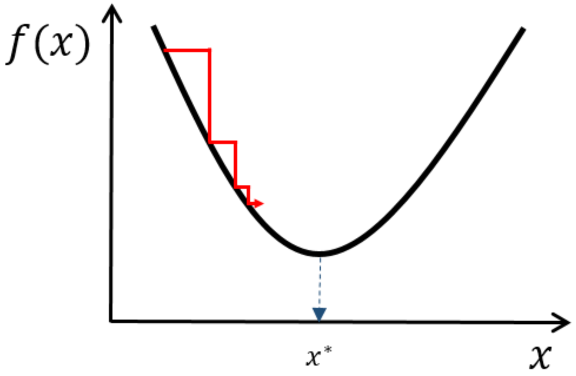
- Learning rate



Too small: converge very slowly

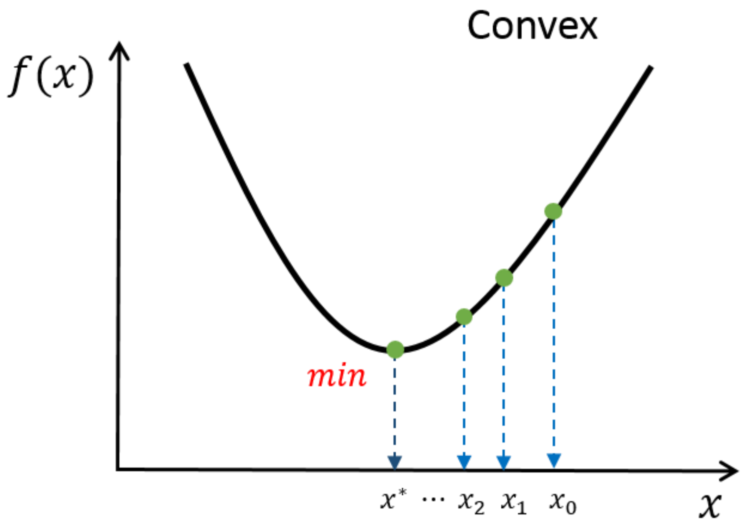


Too big: overshoot and even diverge

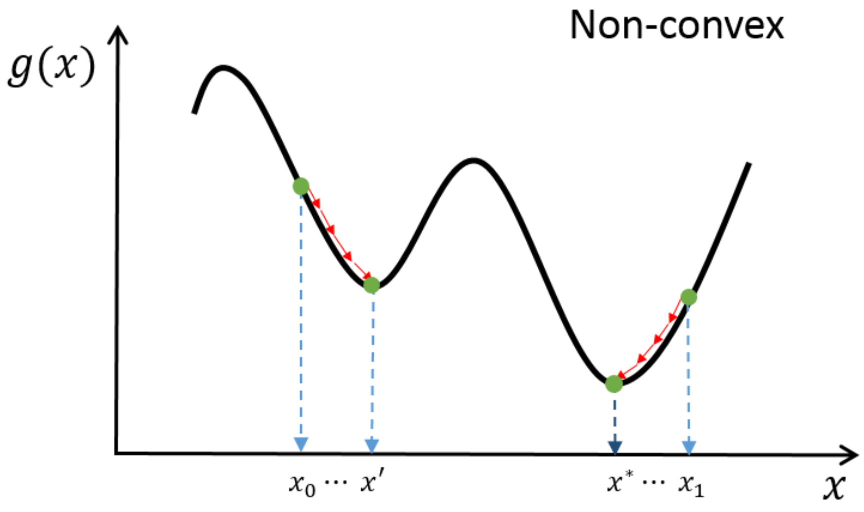


Reduce size over time

Where will We Converge?



Any local minimum is a global minimum



Multiple local minima may exist

- Random initialization
- Multiple trials

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

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

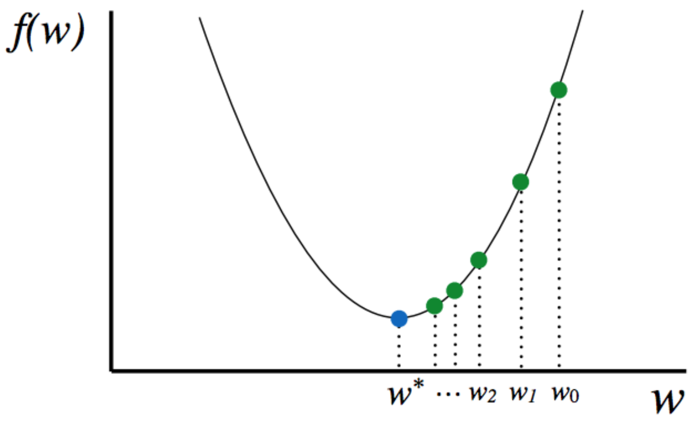
- CVX (or CVXPY) as an optimization solver
  - Only for convex problems
  - Download: https://www.cvxpy.org/

- Gradient descent
  - Neural networks/deep learning
  - TensorFlow

Summary: Training Neural Networks

- Optimization procedure

Start at a random point
Repeat
    Determine a descent direction
    Choose a step size
    Update
Until stopping criterion is satisfied



- 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