# Optimization for Machine Learning

# Optimization for machine learning

- Many engineering disciplines cannot survive without optimization
- Including machine learning
- The generic ERM + regularization minimization:

$$\min_{\boldsymbol{w}} \sum_{i}^{n} L(f_{\boldsymbol{w}}(x_i), y_i) + \Omega(\boldsymbol{w})$$

Minimize a sum of loss function on every training example

# How to solve optimization problems

$$\min_{\boldsymbol{w}} f(\boldsymbol{w})$$

$$\nabla f(\boldsymbol{w}) = \left[\frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}, \dots, \frac{\partial f}{\partial w_d}\right]^{\top}$$

- First-order condition (Stationarity):

$$\nabla f(\boldsymbol{w}) = 0$$

- *Necessary* for optimality
  - Not *sufficient*!
  - Sufficient when $f(\boldsymbol{w})$ convex (will talk about later)

# Try take some gradients

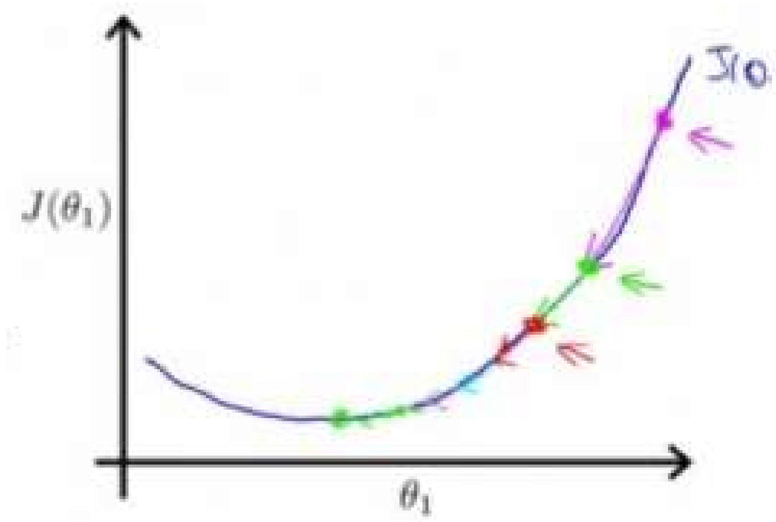
- The gradient of $\boldsymbol{w}^\top \boldsymbol{x}$ w.r.t. to $\boldsymbol{w}$?
- The gradient of $\left(\boldsymbol{w}^\top \boldsymbol{x} - y\right)^2$ w.r.t. to $\boldsymbol{w}$?

# Gradient Descent

$$\min_{w} f(w)$$

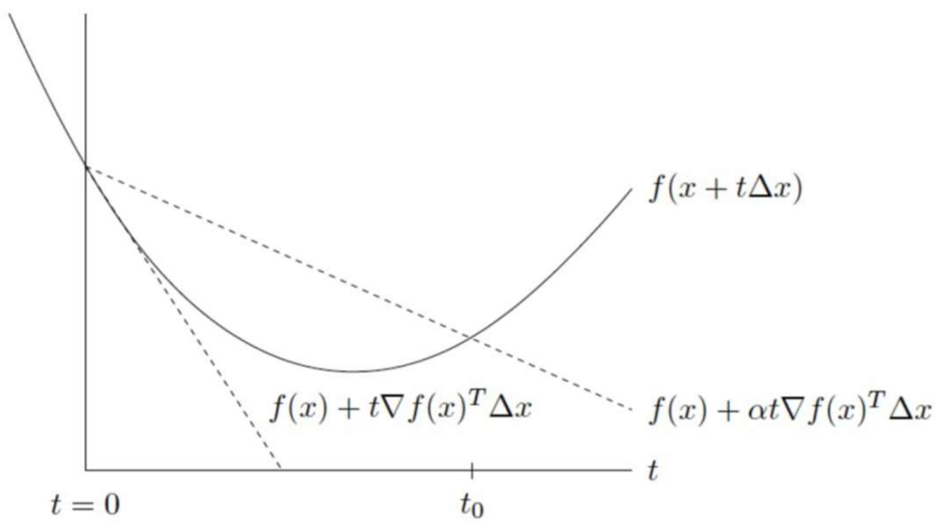
while $\|\nabla(w)\| > \epsilon$
  $w = w - \alpha \nabla f(w)$

$\alpha$: Step size (Learning rate)



# Line search, step size

- One needs the correct step size to converge faster
- In traditional optimization, in order to decide step-size, line search was often used on the descent direction
  - Satisfy certain conditions (e.g. Armijo-Goldstein, Frank-Wolfe)



# Gradient direction can be bad

- $\min_{w_1, w_2} (w_1 - 1)^2 + 100(w_2 - 1)^2$

- What is $\nabla f(\boldsymbol{w})$?
- What is $\nabla f(\boldsymbol{w})$ at (0,0)?

- What is a good step size?

- That's why usually need second-order information
  - Curiously deep learning does not often use second-order information

# Hessian

- The Hessian: $\mathbf{H} = \nabla^2 f = \begin{pmatrix} \partial^2 f / \partial w_1^2 & \cdots & \partial^2 f / \partial w_1 \partial w_d \\ \vdots & \ddots & \vdots \\ \partial^2 f / \partial w_d \partial w_1 & \cdots & \partial^2 f / \partial w_d^2 \end{pmatrix}$

- A second-order Taylor expansion:

$$f(\boldsymbol{w}) = f(\boldsymbol{a}) + \nabla_w f(\boldsymbol{a})(\boldsymbol{w} - \boldsymbol{a}) + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{a})^\top \mathbf{H}(\boldsymbol{a})(\boldsymbol{w} - \boldsymbol{a}) + o(||\boldsymbol{w} - \boldsymbol{a}||^2)$$

$\min_w f(w)$

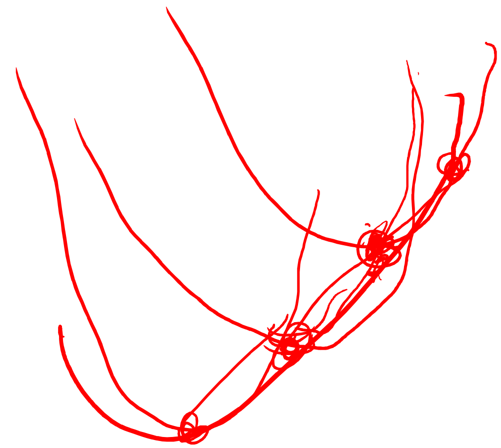
$\nabla_w f(a) + H(a)(w-a) = 0$

# Newton direction

$$d = \left[\nabla^2 f(w)\right]^{-1} \nabla f(w)$$

- e.g. $\min_{w_1, w_2} (w_1 - 1)^2 + 100(w_2 - 1)^2$
- Algorithm:

$$\text{while } \|\nabla f(w)\| > \epsilon$$
$$w = w - \alpha d$$



$$H = \nabla^2 f(w) = \begin{bmatrix} 2 & 0 \\ 0 & 200 \end{bmatrix}$$

$$H^{-1} = \begin{bmatrix} 1/2 & 0 \\ 0 & 1/200 \end{bmatrix}$$

$$\nabla f(w)\big|_{(0,0)} = \begin{bmatrix} -2 \\ -200 \end{bmatrix}$$

- Other variants of Newton-type methods:
  - Quasi-Newton (e.g. BFGS, use an approximation of Hessian)
  - Limited Memory Quasi-Newton (use a low-rank Hessian)
  - Barzilai-Borwein (use diagonal of Hessian)

$$d = [\nabla^2 f(w)]^{-1} \nabla f(w)$$

$$H^{-1} = \sum_{i=1}^{k} \alpha_i h_i h_i^T$$

$$H^{-1} \nabla f(w) = \sum_{i=1}^{k} \alpha_i h_i \left[h_i^T \nabla f(w)\right]$$

Limited-memory Newton method

LBFGS

# Convexity

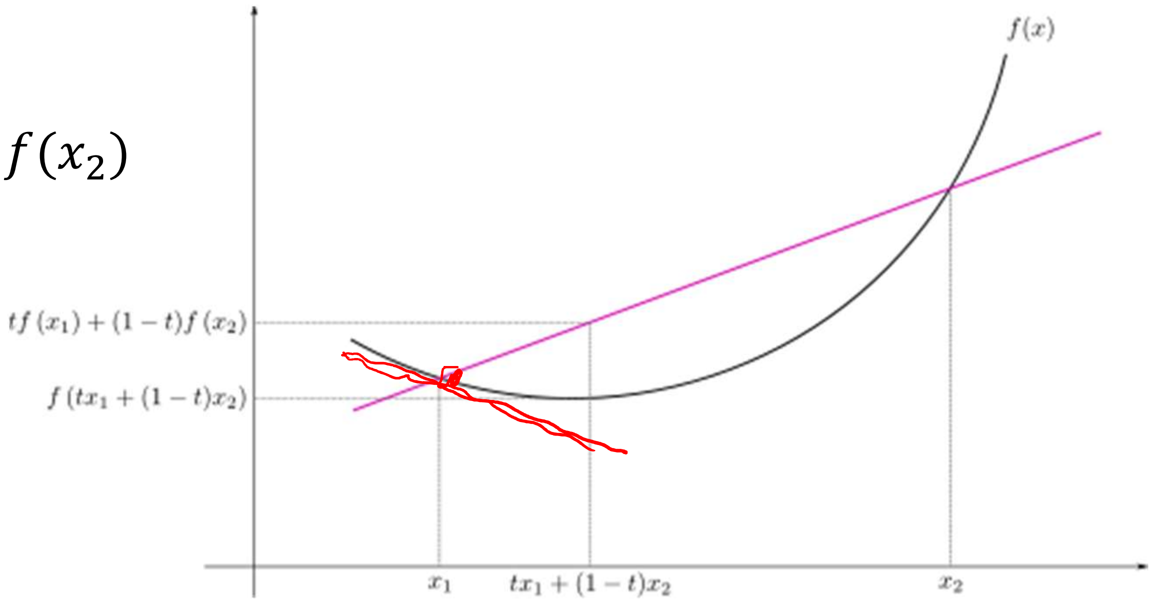
- F is convex if

$$\forall x_1, x_2 \in X, \forall t \in [0,1],$$
$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$$

- First order condition:

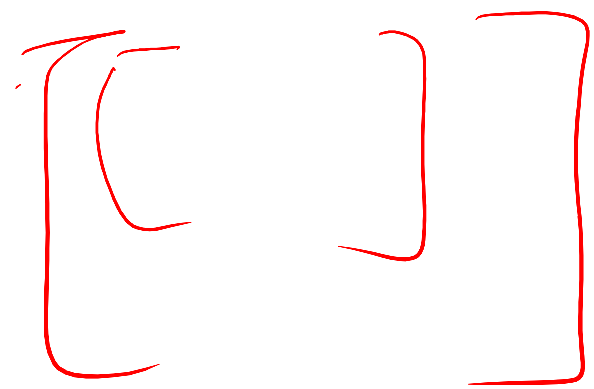
$$f(y) \geq f(x) + \nabla f(x)^\top (y - x).$$

- Second order condition:
- $\nabla^2 f(x) \geq 0$



# Positive semi-definiteness review

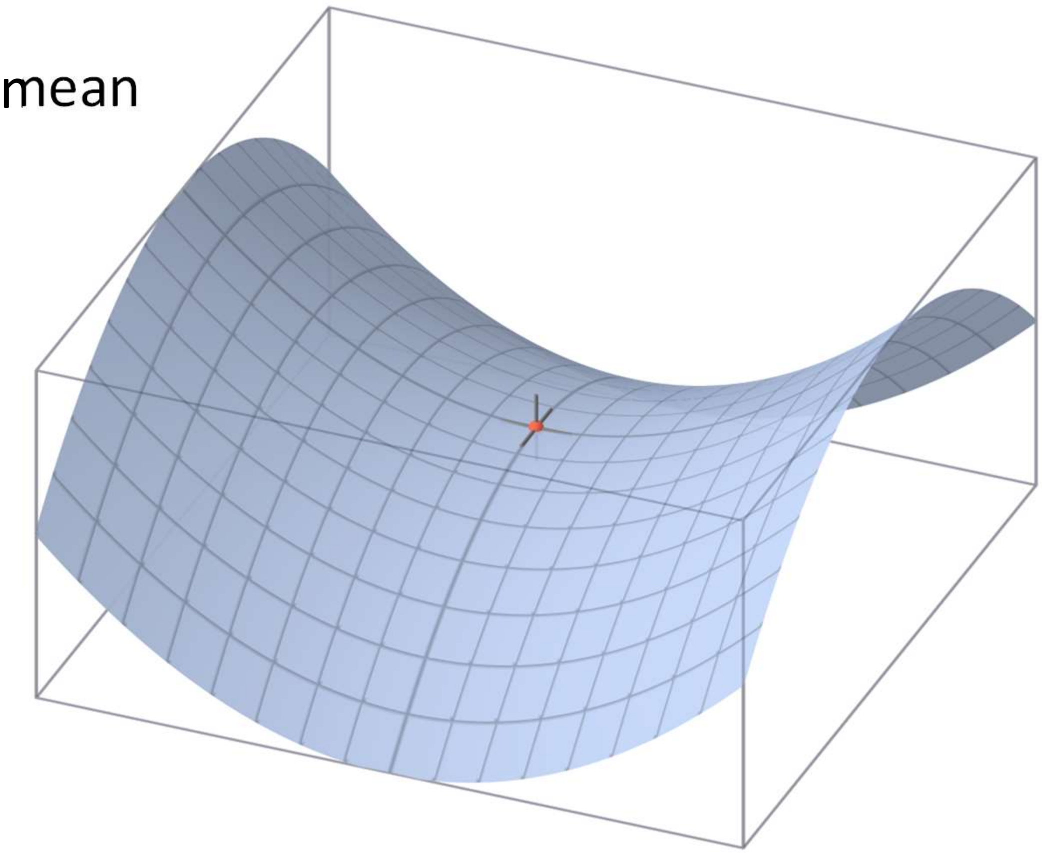
- Important concept in linear algebra
- $\mathbf{M}\ p.s.d. \Leftrightarrow \mathbf{z}^{\top}\mathbf{M}\mathbf{z} \geq 0$
- All eigenvalues of $\mathbf{M}$ are nonnegative
- All principal minors are nonnegative



- Positive-definiteness:
  - (Change >=0 to >0)

# Saddle Point

- Stationarity doesn't necessarily mean local optimum
  - Simple example: $z = x^2 - y^2$
  - $x = 0, y = 0$
- Definition of local optimum
  - Stationary + Locally (strongly) convex



# Nonconvexity