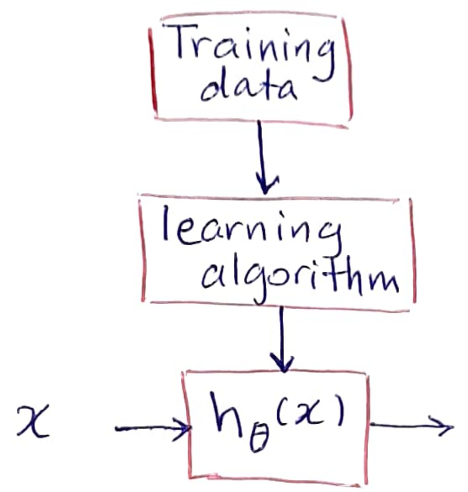
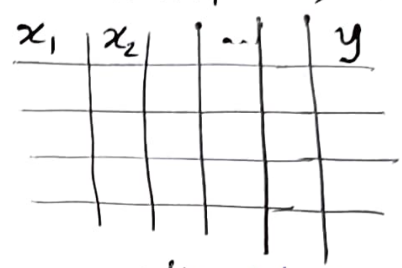
→ Numerical Optimization for ML, session 4, Mans.

AI45, 10/2/2025

Review



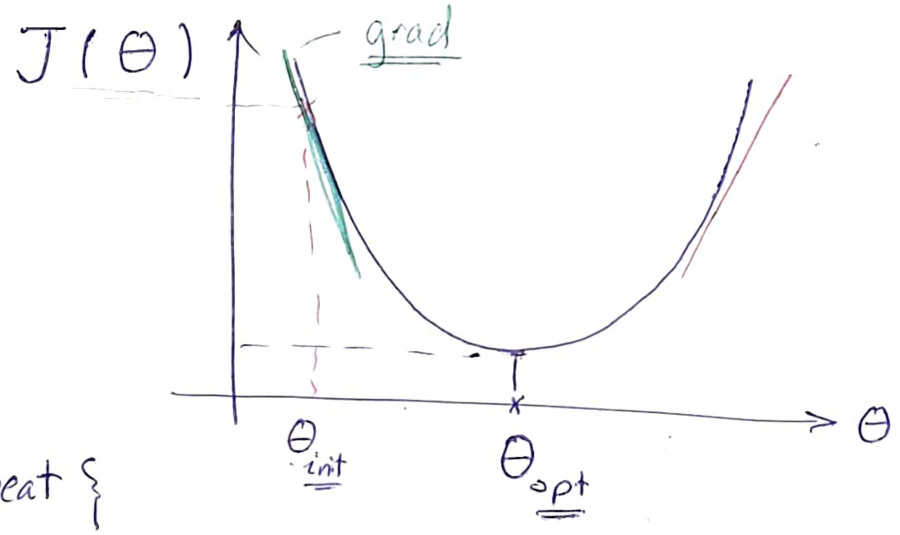
m-data points, n-features



$x_j^{(i)}$ ← data example #i

j ← feature #j

→ Cost function of $(\hat{y} - y)$ "error"



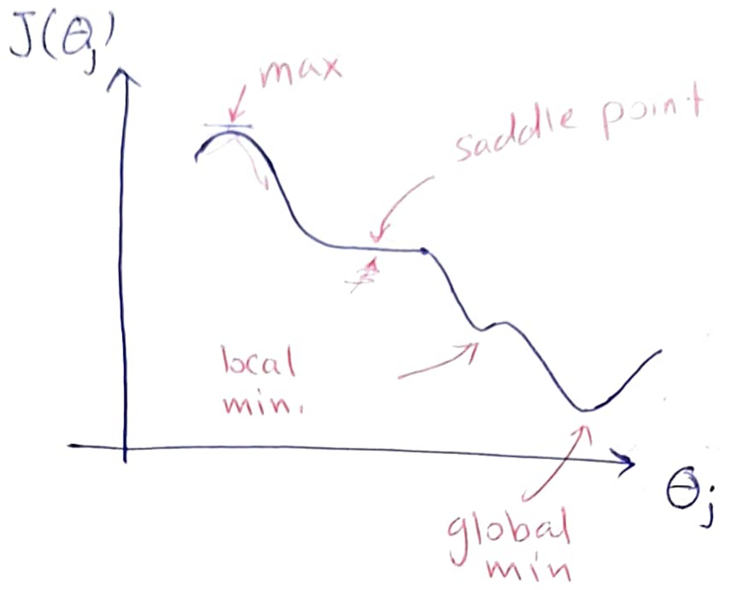
Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \dots \theta_n)$$

assign new value to the parameter θ_j

} until convergence

$$\frac{\partial}{\partial \theta_j} J(\theta) = 0$$



$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\vec{\theta})$$

$$\theta_{j\_new} = \theta_{j\_old} - \alpha \frac{\partial}{\partial \theta_j} J(\vec{\theta})$$

for any θ_j

(note that we update all θ_j's simultaneously)

n-features ; $x_1, x_2, \cdots, x_n$

$\Rightarrow$ (n+1)-parameters $\theta_0, \theta_1, \cdots, \theta_n$

e.g., linear regression

$$h_{\vec{\theta}}(\vec{x}) = \theta_0 \underbrace{x_0}_{x_0 = 1} + \theta_1 x_1 + \cdots + \theta_n x_n$$

e.g., logistic regression

$$h_{\vec{\theta}}(\vec{x}) = \frac{1}{1 + e^{-(\theta_0 \underbrace{x_0}_{x_0=1} + \theta_1 x_1 + \cdots \theta_n x_n)}}$$

$$\vec{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \vec{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$\vec{\theta} := \vec{\theta} - \underline{\underline{\alpha}} \, \nabla(J(\vec{\theta}))$$

$$\vec{\theta}^{(t+1)} = \vec{\theta}^{(t)} - \underline{\alpha} \, \nabla(J(\vec{\theta}^{(t)}))$$

$$\vec{\theta}_{K+1} = \vec{\theta}_K - \underline{\alpha} \, \nabla(J(\vec{\theta}_K))$$

→ Vanilla gradient descent

$$J(\theta) = \frac{1}{\underline{2m}} \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)})^2$$

for all data points m

→ Variants :

- SGD
- mini-batch GD

<u>accelerating GD</u>

<u>Momentum-based methods</u> → Momentum GD

→ Nesterov method

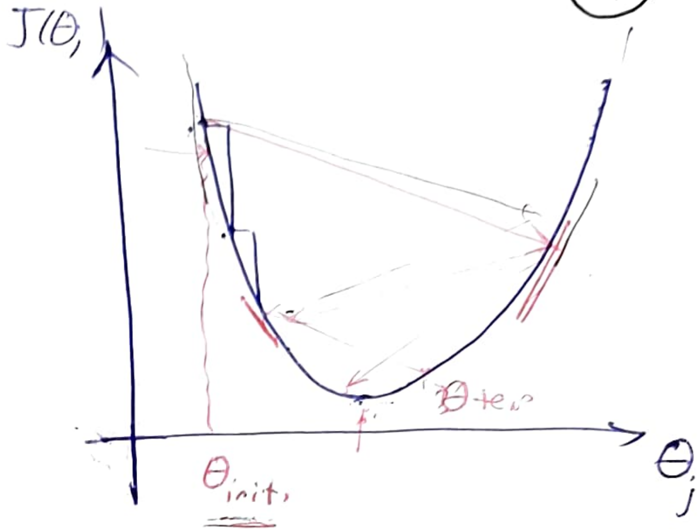
- Momemntum GD

$\cancel{v(t)} =$

$$v^{(t)} = \underline{\beta}\, v^{(t-1)} + \alpha \nabla J(\theta^{(t)})$$

γ

$$\theta^{(t+1)} = \theta^{(t)} - v^{(t)}$$



1. $\theta^{(1)} = \theta_{init}$

$v^{(1)} = 0 + \alpha \nabla J(\theta^{(1)})$

$\theta^{(2)} = \theta^{(1)} + v^{(1)}$

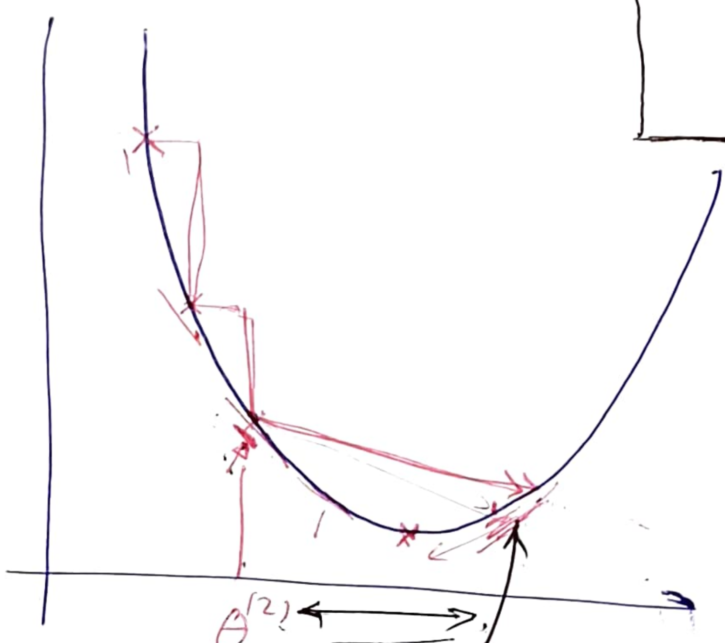
$$\boxed{\theta^{(2)} = \theta^{(1)} - \alpha \nabla J(\theta^{(1)})}$$

2. $v^{(2)} = \beta\, v^{(1)} + \alpha \nabla J(\theta^{(2)})$

$v^{(2)} = \beta(\alpha \nabla J(\theta^{(1)}) + \alpha \nabla J(\theta^{(2)}))$

$\theta^{(3)} = \theta^{(2)} - \underline{\qquad\qquad}$

exponentially decaying weighted average



Nesterov's ?

~~expected~~ Predicted (extrapolated) new θ

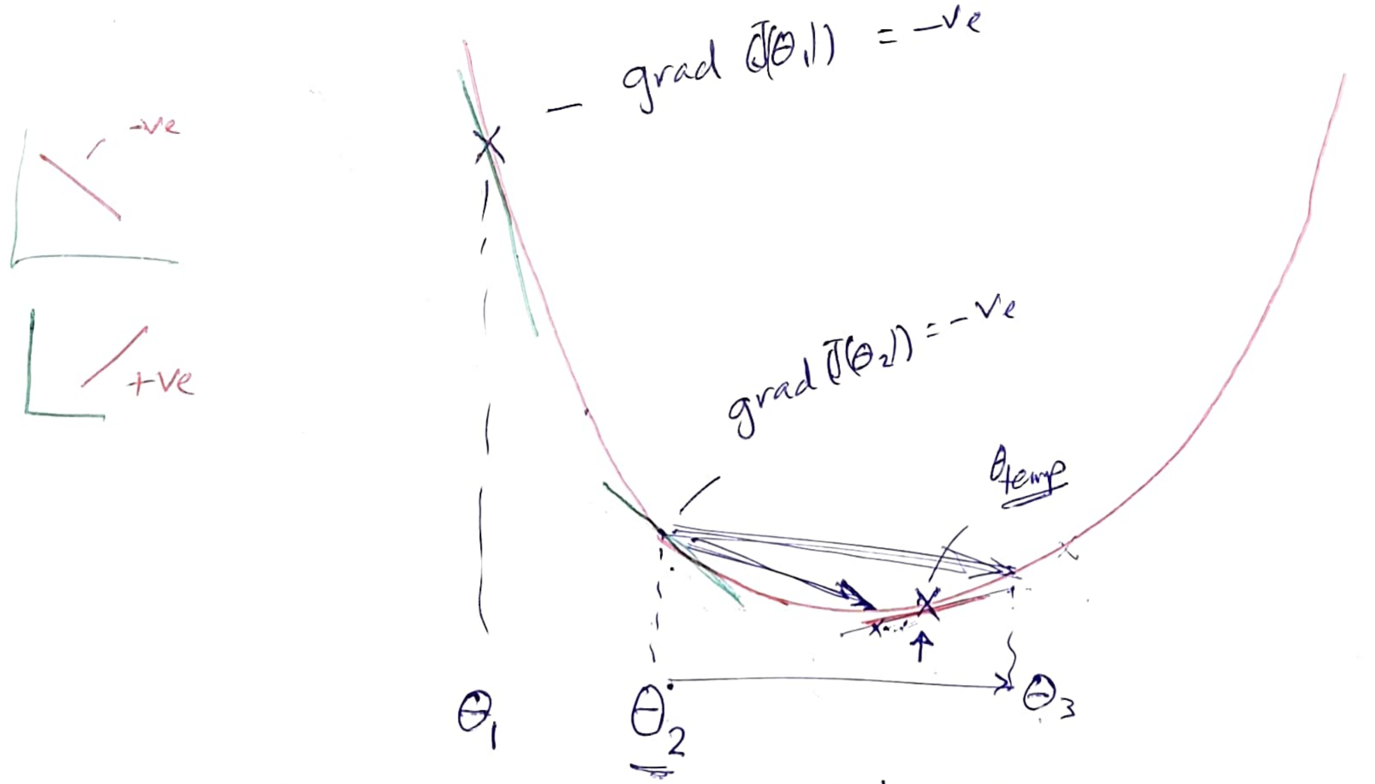
θ^{t+1} (temp)

Nesterov method ;

$$v^{(t)} = \beta v^{(t-1)} + \alpha \nabla J(\underbrace{\theta^{(t)} - \beta v^{(t-1)}}_{\theta_{temp}})$$

$$\theta^{(t+1)} = \theta^{(t)} - v^{(t)}$$

$$= \boxed{\underbrace{\theta^{(t)} - \beta v^{(t-1)}}_{\theta_{temp}}} - \alpha \nabla J(\theta_{temp})$$



momentum	Nesterov
$v(t) = \beta\,(\underbrace{grad(J(\theta_1))}_{-ve} + \underbrace{grad(J(\theta_2))}_{-ve})$	$v(t) = \beta\,(\underbrace{grad(J(\theta_1))}_{-ve} + grad(\theta_{temp}))$
$= -ve \uparrow$	$\theta_{temp} = \theta_2 - \beta\, grad(J(\theta_1))$
$\theta_3 = \theta_2 - (-ve \uparrow)$	$\theta_{temp} >> \theta_2$
$\theta_3 >> \theta_2$	$\theta_3 = \theta_2 - (-ve/+ve) \downarrow$

Accelerating GD

→ learning rate

$$\nabla J = \begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \frac{\partial J}{\partial \theta_1} \\ \vdots \\ \frac{\partial J}{\partial \theta_n} \end{bmatrix}$$

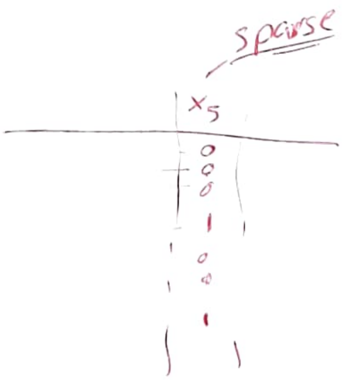
$$\vec{\theta} := \vec{\theta} - \underbrace{\alpha}_{\text{learning rate}} \nabla J(\vec{\theta})$$

- Can we use different α for each parameter?

$$\vec{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \vec{\alpha} = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \alpha_j \frac{\partial}{\partial \theta_j} J(\vec{\theta}^{(t)})$$

for $\vec{\alpha}$



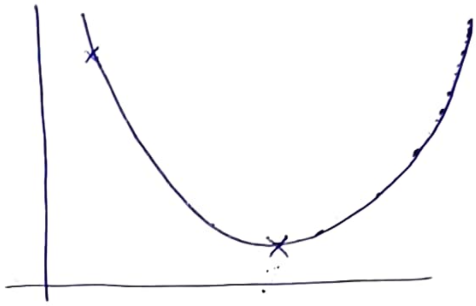
$$\vec{\theta} := \underbrace{\vec{\theta}}_{\text{vector}} - \underbrace{\vec{\alpha}}_{\text{vector}} \odot \underbrace{\nabla J(\vec{\theta})}_{\text{vector} \in \mathbb{R}^{n+1}}$$

Hadamard product (element-wise)

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \quad \vec{x} \odot \vec{y} = \begin{bmatrix} x_1 y_1 \\ x_2 y_2 \\ x_3 y_3 \end{bmatrix}$$

→ Decaying learning rate

Starting with large α,
decrease α with time.



I- adptive gradient (Adagrad)

$$\theta_j^{(t+1)} = \theta_j^{(t)} - \frac{\alpha_j}{\varepsilon + \sqrt{\sum_{k=1}^{t} \left(grad(\theta_j^{k})\right)^2}}\, grad(\quad)$$

numerical stabilizer

expressed with velocity term $v^{(t)} = v^{(t-1)} + (grad(\ ))^2$

$$\theta^{(t+1)} = \theta^{(t)} - \frac{\alpha}{\varepsilon + \sqrt{v^{(t)}}}\, grad(\quad)$$

II- RMSProp (root mean square propagation)

adaptive learning rate

$$\Theta^{(t+1)} = \Theta^{(t)} - \boxed{\frac{\alpha}{\varepsilon + \sqrt{\nu^{(t)}}}} \, \text{grad}(\Theta^{(t)})$$

$$\nu^{(t)} = \beta \, \nu^{(t-1)} + (1-\beta)(\text{grad}(\;))^2$$

III - Adam (Adaptive momentum)