



Machine Learning

Stochastic Gradient Descent

Dr. Mehran Safayani

safayani@iut.ac.ir

safayani.iut.ac.ir



<https://www.aparat.com/mehran.safayani>



https://github.com/safayani/machine_learning_course



Stochastic Gradient Descent

$$L(\theta) = \frac{1}{m} \sum_{i=1}^m L_i(\theta) \quad (\text{cost function}) \quad L_i = (\hat{y}_i - y_i)^2$$

$L_i(\theta)$ = cost of i th training sample

SGD:

$$\theta^{t+1} = \theta^t - \alpha \nabla L_i(\theta^t)$$

$\nabla L_i(\theta^t)$:

یک تخمین از $\nabla L(\theta^t)$ است و محاسباتش کم هزینه تر است.
می توان نشان داد:

$$E[\nabla L_i(\theta)] = \nabla L(\theta)$$

Mini_Batch SGD

$$L = \frac{1}{|B|} \sum_{i \in B} L_i(\theta^t) \quad (B: \text{تعداد})$$

$$\theta^{t+1} = \theta^t - \alpha g \quad g = \frac{dL}{d\theta}$$

- یک مجموعه تصادفی به اندازه $|B|$ از داده های آموزشی انتخاب می کنیم.
- امکان موازی سازی با Mini_Batch SGD بیشتر از SGD است.
- حجم محاسبات : $O(|B|.n)$

GD

```
Repeat{  
     $d\theta = 0$   
    for i = 1 to m:  
        compute  $d\theta^i$   
         $d\theta \ += d\theta^i$   
         $\theta = \theta - \alpha \frac{1}{m} d\theta$   
} until convergence
```

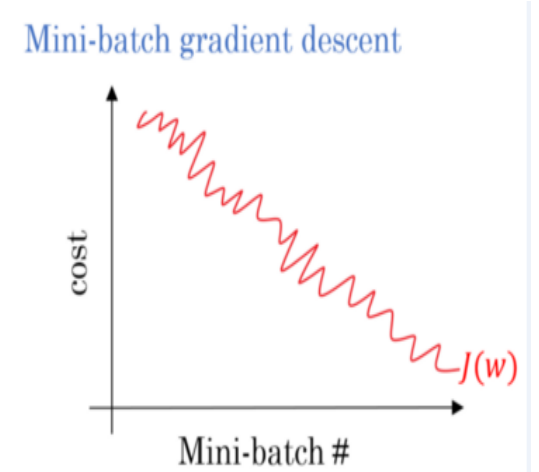
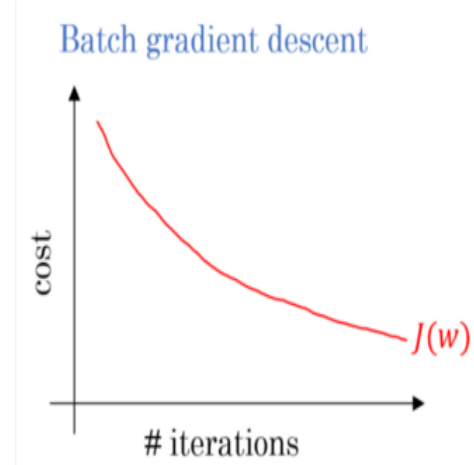
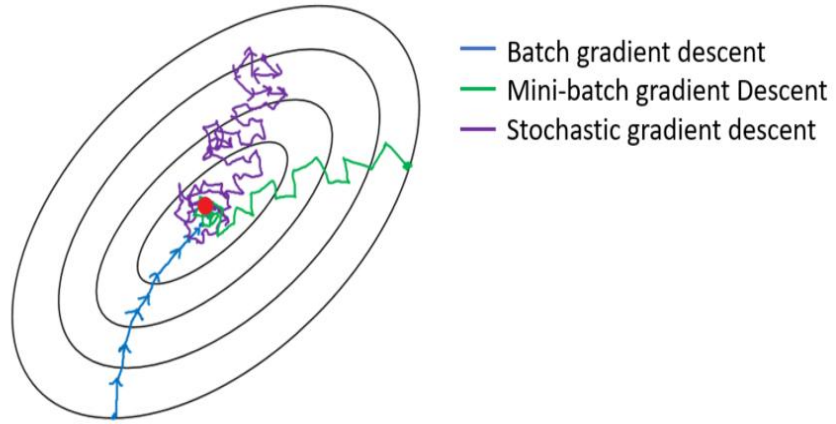
mini_batch SGD

```
 $T = \frac{m}{B}$  ;  $B = \text{batch\_size}$   
  
Repeat{  
    for j= 1 to T:  
         $d\theta = 0$   
  
        for i = 1 to B:  
            compute  $d\theta^i$   
             $d\theta \ += d\theta^i$   
  
         $\theta = \theta - \alpha \frac{1}{B} d\theta$   
}until convergence
```

SGD

```
Repeat{  
    for i = 1 to m:  
        compute  $d\theta^i$   
         $\theta = \theta - \alpha d\theta^i$   
} until convergence
```

Comparison



$$\{x^1, x^2, \dots, x^{1000}\}, \quad x^1 = x^2 = \dots = x^{1000}$$

$$\frac{1}{1000} \sum_{i=1}^{1000} L(\hat{y}_i, y_i) = \frac{1}{1000} * 1000 L(\hat{y}_i, y_i)$$

$$\left. \begin{matrix} x^1 \\ x^2 \\ \dots \\ x^{1000} \end{matrix} \right\} \theta^1 \text{ updated}$$

$$\left. \begin{matrix} x^1 \\ x^2 \\ \dots \\ x^{1000} \end{matrix} \right\} \theta^2 \text{ updated}$$

$$\left. \begin{matrix} \dots \\ x^1 \\ x^2 \\ \dots \\ x^{1000} \end{matrix} \right\} \theta^{1000} \text{ updated}$$

$$\theta^i = \theta^{*i}$$

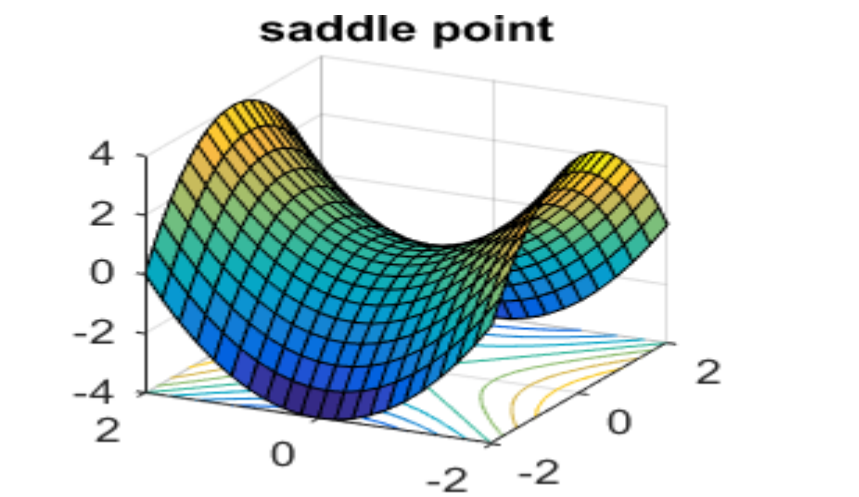
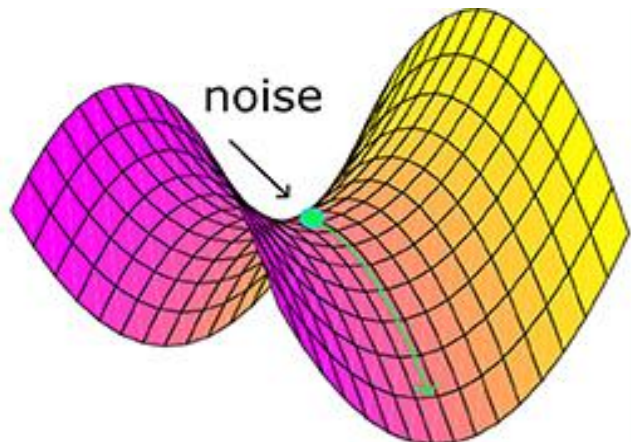
$$\equiv$$

1
epoch

SGD:

$$\left. \begin{matrix} x^1: \theta^{*1} \text{ updated} \\ x^2: \theta^{*2} \text{ updated} \\ x^3: \theta^{*3} \text{ updated} \\ \vdots \\ x^{1000}: \theta^{*1000} \text{ updated} \end{matrix} \right\}$$

Saddle Points



به کمک **SGD** میتوان از نقطه زین اسبی فرار کرد

Comparison

SGD

سریع بودن تکرار
استفاده از افزودگی یا همبستگی در داده ها
موازی سازی نامناسب
تابع خطا خیلی نویزی است
کند بودن الگوریتم
سرعت الگوریتم کاهش می یابد.
خیلی نویزی

Mini-batch SGD

یک مصالحه بین روش
Batch GD و SGD

Batch GD

تابع خطا خیلی نرم کاهش می یابد
موازی سازی مناسب
طولانی بودن هر تکرار
ماتریس های خیلی بزرگ
گیرکردن در نقاط زین اسبی
کند بدلیل عدم استفاده مناسب از
افزودگی یا همبستگی در داده ها

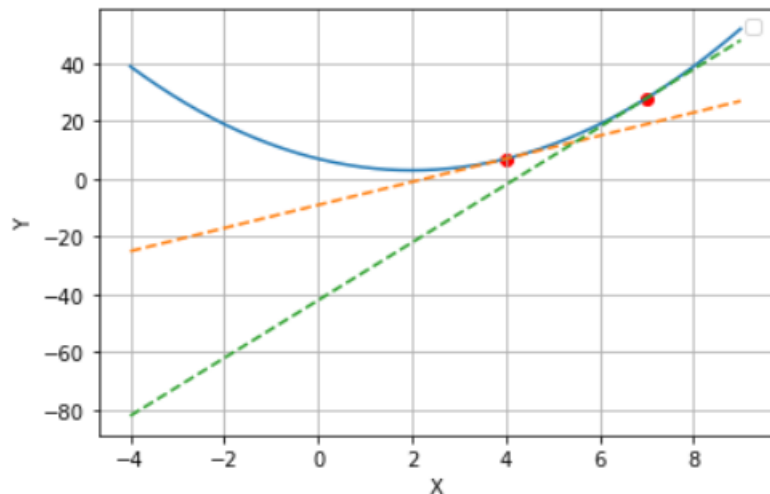
$m \leq 2000$: Batch

Mini_Batch: 64 , 128, 256 , 512

Subgradient Method

این روش برای توابعی که در برخی نقاط مشتق پذیر نیستند بکار می رود.

برای توابع محدب مشتق پذیر داریم: $\mathcal{L}(u) \geq \mathcal{L}(w) + \nabla \mathcal{L}(w)^T (u - w) \quad \forall u, w$



بدین معنی که تابع همیشه بزرگتر از تخمین خطی اش است

Subgradient Method

Subgradient:

A vector $\mathbf{g} \in \mathbb{R}^D$ such that

$$\mathcal{L}(\mathbf{u}) \geq \mathcal{L}(\mathbf{w}) + \mathbf{g}^\top (\mathbf{u} - \mathbf{w}) \quad \forall \mathbf{u}$$

is called a **subgradient** to the function \mathcal{L} at \mathbf{w} .

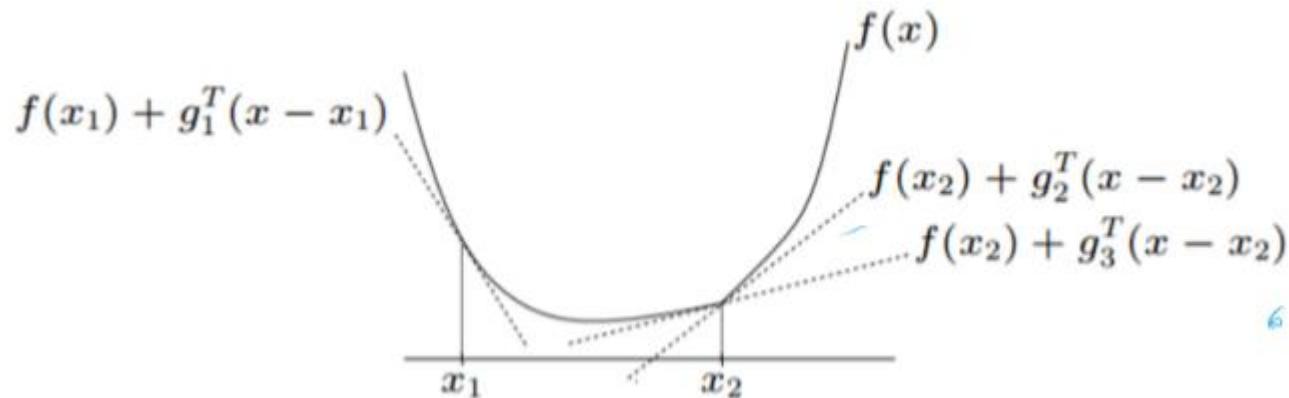
اگر تابع $L(w)$ مشتق پذیر باشد $g = \Delta L(w)$ است.

Subgradient Descent:

$$\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} - \gamma \mathbf{g}$$

Subgradient Method

$$|x| \longrightarrow g = \begin{cases} 1 & x > 0 \\ [-1, 1] & x = 0 \\ -1 & x < 0 \end{cases}$$



$$g(x_i) = [g_3, g_2]$$

MAE Gradient Descent

- $h_{\theta}(x_i) = \theta_0 + \theta_1 x_i$
- $J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^m |h(x_i) - y_i|$

- $sign(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x \geq 0 \end{cases}$

$$\frac{dJ(\theta_0, \theta_1)}{d\theta_0} = \frac{1}{m} \sum_{i=1}^m sign(h_{\theta}(x_i) - y_i)$$

$$\frac{dJ(\theta_0, \theta_1)}{d\theta_1} = \frac{1}{m} \sum_{i=1}^m sign(h_{\theta}(x_i) - y_i) x_i$$