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

Department of Electrical and computer engineering, Isfahan university of technology, Isfahan, Iran

Stochastic Gradient Descent

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

$L_i(\theta)$ = cost of ith training sample

SGD:

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

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

یک تخمین از ∇L (θ^t) است و محاسباتش کم هزینه تر است.

می توان نشان داد:

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

Mini_Batch SGD

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

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

- 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 = 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{

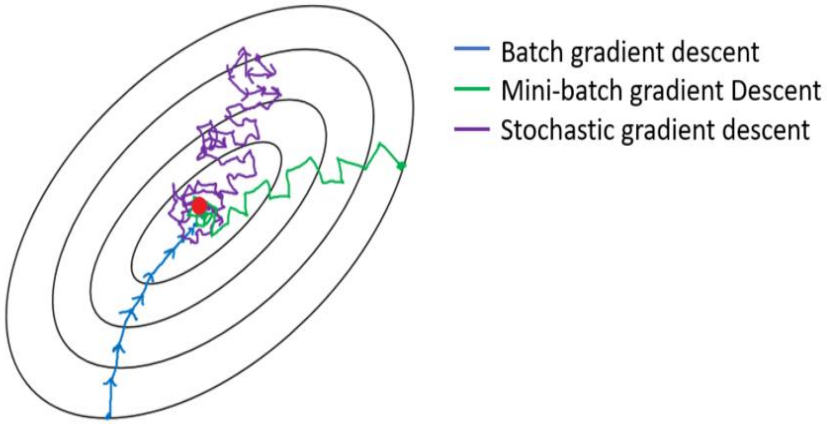
for i = 1 to m:

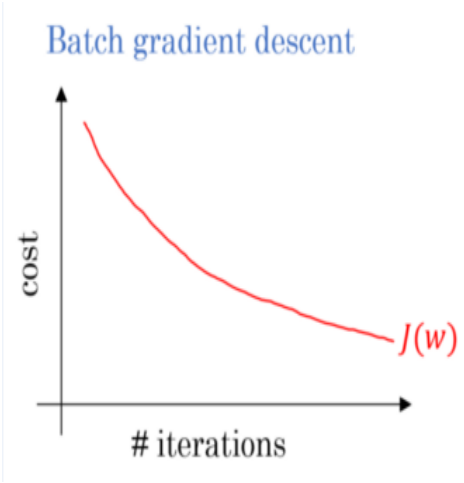
compute $d\theta^i$

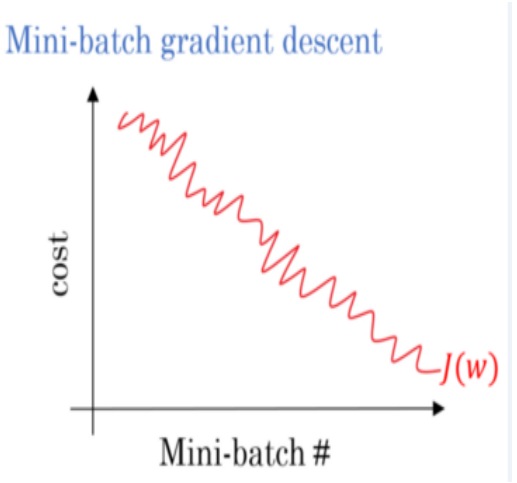
$\theta = \theta - \alpha\, d\theta^i$

} until convergence

Comparison


Batch gradient descent
Mini-batch gradient Descent
Stochastic gradient descent


Batch gradient descent
cost
J(w)
# iterations


Mini-batch gradient descent
cost
J(w)
Mini-batch #

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

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

x^1
x^2
…
x^{1000}
θ^1 updated

x^1
x^2
…
x^{1000}
θ^2 updated

…..
x^1
x^2
…
x^{1000}
θ^{1000} updated

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

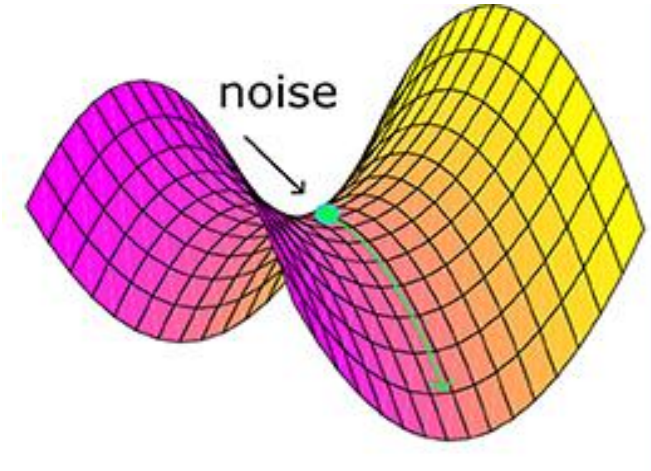
$$\equiv$$

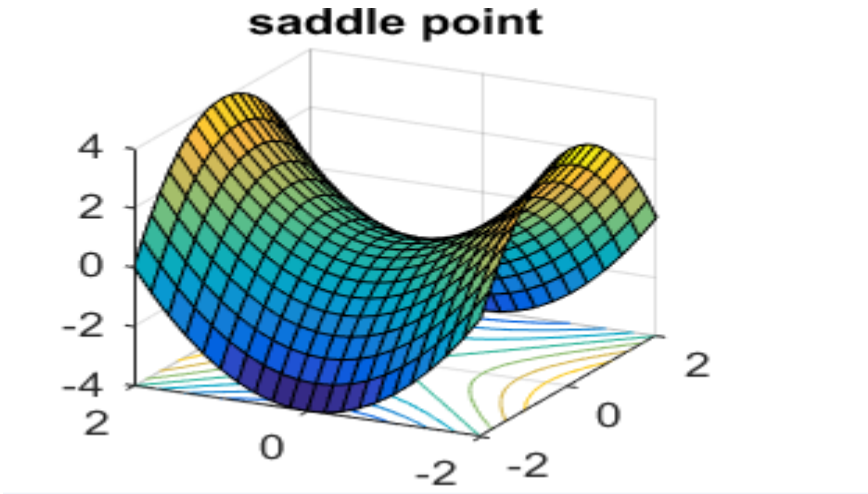
SGD:

1 epoch

x^1: θ^{*1} updated
x^2: θ^{*2} updated
x^3: θ^{*3} updated
.
.
.
x^{1000}: θ^{*1000} updated

Saddle Points





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

Comparison

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
افزونگی یا همبستگی در داده ها

Mini-batch SGD

یک مصالحه بین روش
SGD و Batch GD

SGD

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

$m \leq 2000$: Batch

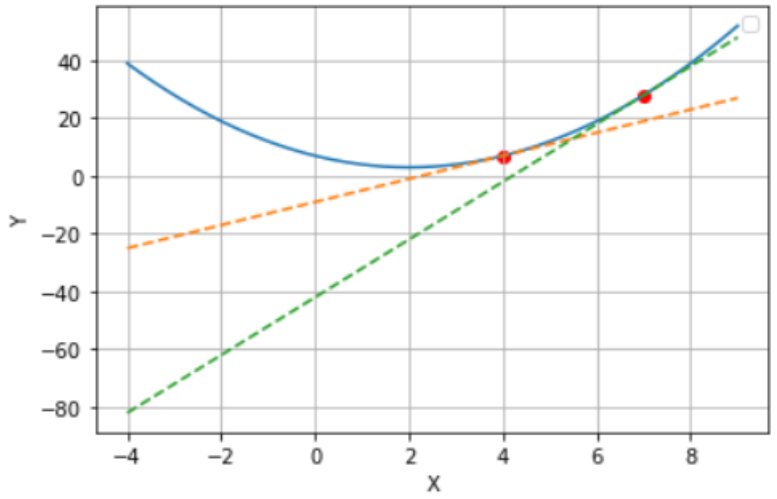
Mini_Batch: 64 , 128, 256 , 512

Subgradient Method

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

برای توابع محدب مشتق پذیر داریم:

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



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 $\mathbf{w}$.

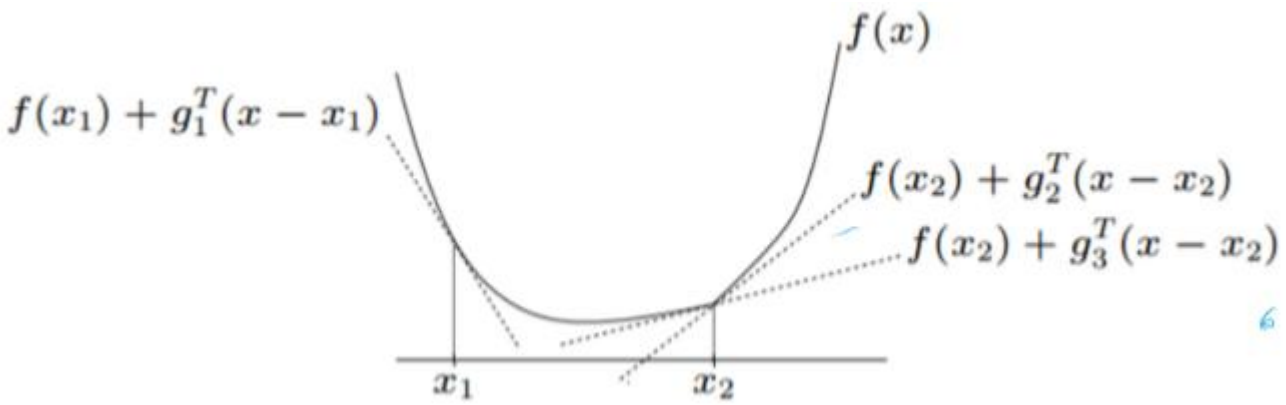
اگر تابع L(w) مشتق پذیر باشد $g = \Delta L(w)$ است.

Subgradient Descent:

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

Subgradient Method

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



$$\mathbf{g}(x_i) = [g_3, g_2]$$