موضوعات منتخب در فیزیک آماری

Session 2 :

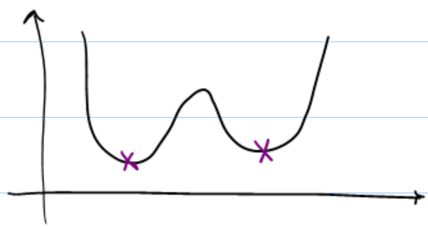
1403, 11, 30

هر مسئلهٔ optimization : $\vec{x} = (\alpha, \beta, \gamma, \dots)$

$$\min[h(\vec{x})]$$

مثال :

$$f(x) = x^4 - x^2$$



$$\vec{x} = \vec{x} + \alpha \vec{u}$$

بهترین مسیر برای صفر کردن مشتق : $\vec{u} = -\vec{\nabla} h(\vec{x})$

* چند نکته مهم :

I) x اولیه که باشد ⇐ جواب ها را تغییر می دهد. $\vec{x}_0 = ?$

II) α که ظاهر می‌شود معمولاً $10^{-2,-3}$ $\vec{x}_{new} = \vec{x} - \alpha \vec{\nabla}_n h(\vec{x})$

III) tolerance که نزدیک بودن x_{new} و x را نشان می دهد

IV) iteration step ⇐ n = step تعداد قدم ها تا min

پارامترها هرچه می توانند باشند، مثلا x ها می توان وزن نورون ها باشد.

- Batch Gradient Desent (single or Mini batch):

برای مثال ۲ الگوریتم به ما می‌دهند و می‌نویسیم regression را :

(x_i, y_i)	
x_1	y_1
x_2	y_2
$\vdots$	$\vdots$
x_m	y_m

$$h(\theta_1, \theta_2) = \frac{1}{2m} \sum_{i=1}^{m} \left[y_i - (\theta_0 + \theta_1 x) \right]^2$$

number of trading sample : m

input - output pairs : x_i, y_i

θ_0 , θ_1 : پارامترهای Optimization

$$\begin{cases} \dfrac{\partial h}{\partial \theta_0} = \dfrac{-1}{m} \displaystyle\sum_{i=1}^{m} y_i - (\theta_0 + \theta_1 x_i) \\ \dfrac{\partial h}{\partial \theta_1} = \dfrac{-1}{m} \displaystyle\sum_{i=1}^{m} x_i \left[y_i - (\theta_0 + \theta_1 x_i) \right] \end{cases}$$

$$\theta_0 = \theta_0 - \alpha \frac{\partial h}{\partial \theta_0} \quad و \quad \theta_1 = \theta_1 - \alpha \frac{\partial h}{\partial \theta_1}$$

حال با توجه به چند نقطه داده شده به آن Run می‌کنیم.

اگر از کل m ها استفاده کنیم ⇐ Batch Gradient Desent ①

اما در هر update، درست m تا جمع می‌کنیم ⇐ برای محاسبه راحت‌تر یکی را انتخاب می‌کنیم.

② single batch gradient desent ⇐ افت و خیز دارد

چون انتخاب آن بدیهی Random است ← stochastic است.

(3) mini batch gradient desent ⇐ حال این تعداد T تا برداریم

معمولاً T را 32 یا 68 می‌گیرند. ⇐ parallel ⇐ GPU می‌توان برد

حال چگونه T تا را انتخاب کنیم:

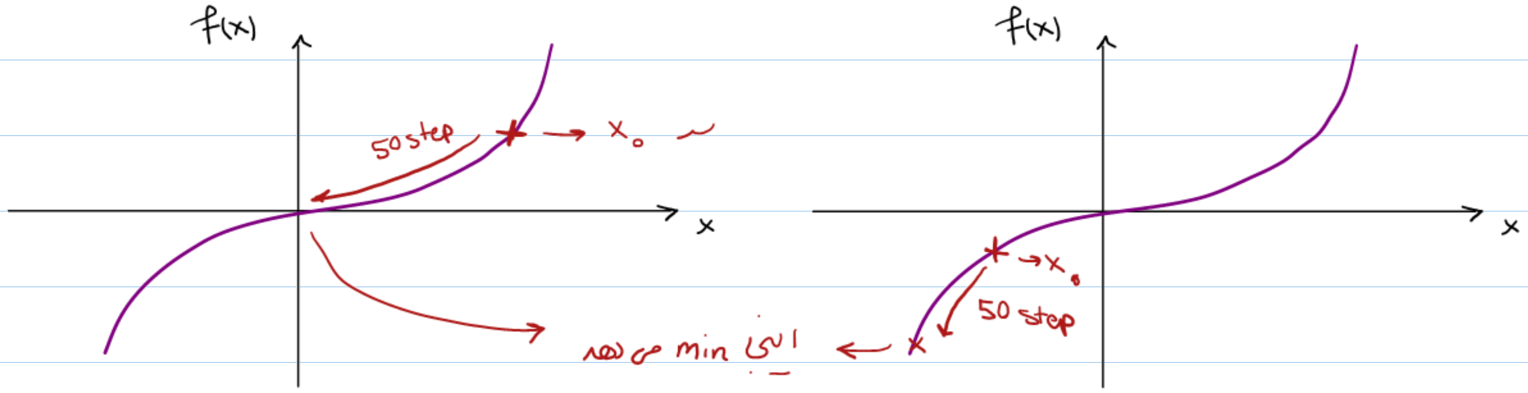
- chunk برمی‌داریم ⇐ پشت سرهم
- Random برمی‌داریم ⇐ Stochastic

مثال: تابع Convex یا غیر Convex چیست؟

این اطلاعات از تابع Convex باشد و اطلاعاتش را داریم ⇐ توان ها ساختاری کمک می‌خواهد.

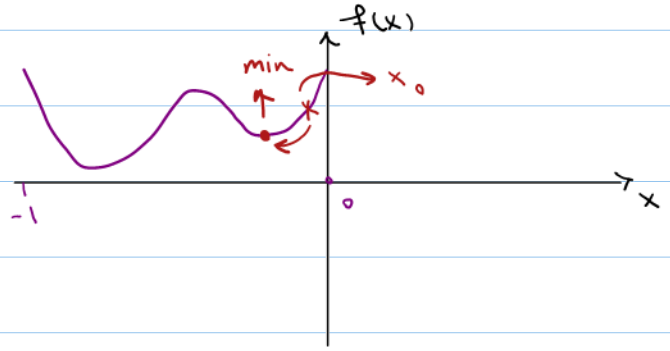
حال این Non-Convex باشد چند حالت داره.

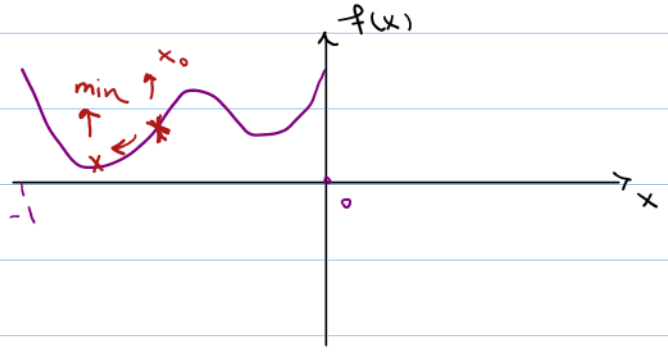
Sattle point: $f(x) = x^3$ $\quad f'(x) = 3x^2$ (مشتق ۰ است اما min نیست)



این واضحاً اشتباه است و غیرترانه

$f(x) = x^3 + \sin(10x)$





=> این که حواسمان نباشد شرایط اولیه را خیلی تعیین نکند => min اشتباه پیدا می‌شود.

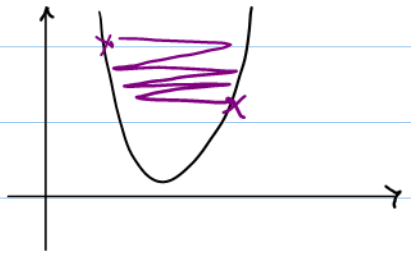
باید این مشکلات را برطرف کنیم.

اما کم کم بیشتر یاد بگیریم.

برای مسائلی که جواب را می‌دانیم. n هزینه Computer من است.

حواسمان در آن زمان سریع‌تر Converge نشد & learning rate (α) را زیاد کنیم.

اما این باعث می‌شود که Converge نشد.



=> باید الگوریتم را عوض کنیم.

Algorithmic variations of GD:

1) Momentum - Steering:

$$x = x - \alpha \underbrace{\nabla_2 h(x)}_{-\vec{u}}$$

داشتیم

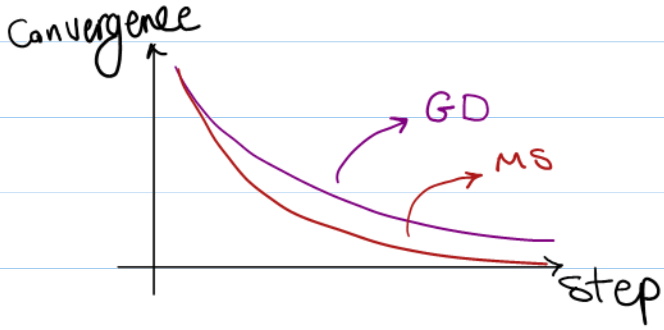
$$x_{s+1} = x_s + \alpha \vec{u}_s$$

حال می‌خواهیم اینجا به u_s از step قبلی اطلاعات بدهیم.

$$u_s = \gamma u_{s-1} - \alpha \nabla_x h(x)$$

γ ↵ typically = 0.9

چون از قبلی سهم می‌برد نام آن momentum است. علاوه به سرعت converge، افت و خیز را هم کاهش می‌دهد.



2) Nesterov Accelerated Gradient (NAS).

$$u_s = \gamma u_{s-1} - \alpha \nabla_x h(x - \gamma u_{s-1})$$

convergence سریع‌تری دارد.

3) Annealing & learning rate schedules

هرچه داریم به Converge نزدیک‌تر می‌شویم، α را کاهش می‌دهیم.

اصطلاحاً می‌گویند سرد کردن ← cooling

چند روش برای کاهش وجود دارد.

3-1) Ada-Grad (Adaptive Gradient Alg).

$$\frac{\partial h}{\partial x_i} \rightarrow g_i^s \qquad g_i^{(0)}, \dots, g_i^{(s-1)}$$

$$G^s = (g_i^0)^2 + \dots + (g_i^s)^2$$

$$x_i = x_i - \frac{\alpha}{\sqrt{G^s + \epsilon}} g_i^s$$

که اگر G صفر شد مشکل ایجاد نکند.

برخی می‌گویند w step قبل را نداریم نه همه را. ⇐ بس آن این است که در w

step نزدیک Convergence، G ها خیلی بزرگ می‌شوند

3-2) RMSprop root mean square propagator:

$$g_s = (\nabla h^2)^{(s)}$$

$$g_s^2 = \beta g_{s-1}^2 + (1-\beta) g_s^2$$

β typically = 0.9

3-3) Ada Delta: an adaptive learning rate method

$$x_i^{s+1} = x_i^s - \frac{RMS[\Delta x_i]^{s-1}}{RMS[g_i]^s} g_i^{(s)}$$

اوایل Δx ها بزرگ هستند ⇐ rate ها بزرگ هستند و کم کم کاهش می‌یابد

4) Adam: adaptive moment estimation

$$g = \nabla_x h$$

putting ∇ as $z \Rightarrow m = \langle z \rangle \quad , \quad v = \langle z^2 \rangle$

$$\begin{cases} m_s = \beta_1 m_{s-1} + (1-\beta_1) g_s & \beta_1 \simeq 0.9 \\ v_s = \beta_2 v_{s-1} + (1-\beta_2) g_s^2 & \beta_2 = 0.999 \end{cases}$$

$$\Rightarrow \begin{cases} \hat{m}_s = \dfrac{m_s}{1-\beta_1^t} \\ \hat{v}_s = \dfrac{v_s}{1-\beta_2^t} \end{cases} \Rightarrow \text{Bias corrector}$$

$$x^{s+1} = x^s - \frac{\alpha}{\sqrt{\hat{v}_s + \epsilon}} m^s$$

$m_0, v_0 = 0 \quad , \quad \alpha \simeq 10^{-3} \quad , \quad \epsilon \simeq 10^{-8}$

Exercise 1: تمرین اول:

دستگاه معادلات خطی $Ax = B$ می‌خواهیم جواب حساب کنیم

می‌خواهیم با optimization حل کنیم. Norm تعریف می‌کنیم

$$h(x) = \min \| Ax - B \|$$

که بتونیم به GD حل کنیم این رو ⊛

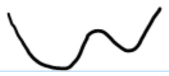
در جایی امکان دارم ماتریس full rank نباشد، یعنی یک سطر ضریب دیگری است.

در اینجا ⇐ ∞ ← inverse det می‌شود.

الان در inverse کردن A دچار مشکل می‌شویم تمام الگوریتم‌ها از کار می‌افتد.

(*) 5 راه‌حل که این مشکل را حل می‌کند بیابید، توضیح دهید. Optimization (*) را برای یکی انجام دهید برای $\det \simeq 10^{-4}$.

حال برای حل مشکل‌تری که گفتیم چند مرحله وجود دارد.

- به صورت uniform در بازه شرایط اولیه رندوم بدهیم ⇐ 
- به u جهت رندوم بدهیم ⇐ شانس داریم برگردد برخلاف گرادیان ⇐ 

شامل:

$n \equiv$ steps $\equiv$ epochs

$\alpha \equiv$ learning rate $= 0.005$

momentum $\gamma = 0.9$

Noise intensity (σ) ⇒ $u = -\nabla h(x) \longrightarrow -\nabla h(x) + \sigma\, \xi(0,1)$

white noise ـِ

Noise decay $= 0.99$ $\quad \sigma^{s+1} = 0.99\, \sigma^{s}$

حال مقادیر را بهترین‌ها را save می‌کنیم

روش‌ها:

روش اول: برای fit کردن معادله خط θ_0, θ_1. (code SGD2)

روش دوم: برای این تابع با حد شرایط اولیه‌ای که داده شده min درست را می‌یابد. (code SGD 3)

برای تست کردن optimization برای چند بعد، می‌توانیم از تست زیر استفاده کنیم

Rosenbrock function چون min در دره بسیار زیر است.

و حواب min و scabble آن را بلد هستیم.

می‌توان برای fine tune کردن پارامترها استفاده کرد.