Stochastic Gradient Descent

Thursday, January 30, 2020 13:09

So far the gradient descent we've seen is

" Batch Gradient Descent "

—— where we use all of in-sample data

there are more types/configurations of gradient descent:

→ "Mini-Batch" GD:

use a subset of total in-sample for gradient descent

⊕ save computation per update

⊖ less accurate

⊕ faster to use many smaller "noisy" GDs than a few accurate GDs

⊕ "noisy" helps with generalization.

"Stochastic gradient descent (SGD)"

extreme case of mini batch: Single sample

$$\underline{w}(t+1) = \underline{w}(t) - \eta_t \nabla e_{n(t)} \cdot (\underline{w}(t))$$

just a single error

the average of individual gradient of error is the actual gradient:

$$\mathbb{E}\left[\nabla e_{n(t)}(\underline{w}(t))\right] = \frac{1}{N}\sum_{n=1}^{N} \nabla e_n(\underline{w}(t))$$

terminology: "Epoch" —— one run of GD through ALL training sample

Stochastic GD: it "hovers" around the minimum (erratic behaviour)

Batch GD

To fix: Have variable learning rate η:

Condition: $\sum_{t=1}^{\infty} \eta_t = \infty$ and $\sum_{t=1}^{\infty} \eta_t^2 < \infty$ guarantees convergence

method ① $\eta_t = (1 - \frac{t}{\tau}) \eta_0 + \frac{t}{\tau} \eta_\tau$, $\eta_\tau \approx 0.01 \eta_0$

$t = 1, 2, 3 \dots \tau$

method ② $\eta_t = \frac{\gamma}{\sqrt{t}}$

method ③ adative learning rate:

learning rate depends on - properties of w
- features
- directions
etc.

To stop:

① Enough iterations

② Error difference threshold
(but computing error will affect computation time)

Aptaptive Learning Rate SGD:

① AdaGrad

→ downscale η_t by squareroot of sum of all historical squares of the gradient

element wise multiplication

gradient est. = $\hat{g}_t$

accumulate = $\underline{r}_t = \underline{r}_{t-1} + \hat{g}_t \otimes \hat{g}_t$

update $\underline{w}(t+1) = \underline{w}(t) - \eta \left(\frac{1}{\delta + \sqrt{\underline{r}_t}} \right) \otimes \hat{g}_t$

prevent $\frac{1}{0}$

element wise

prevent $\frac{1}{0}$

element wise squareroot & division

② RMS Prop

gradient $\underline{\hat{g}}_t$

accumulate $\underline{r}_t = \rho\, r_{t-1} + (1-\rho)\, \hat{g}_t \otimes \hat{g}_t$

"forget" factor ⇒ slows down the shrinkage

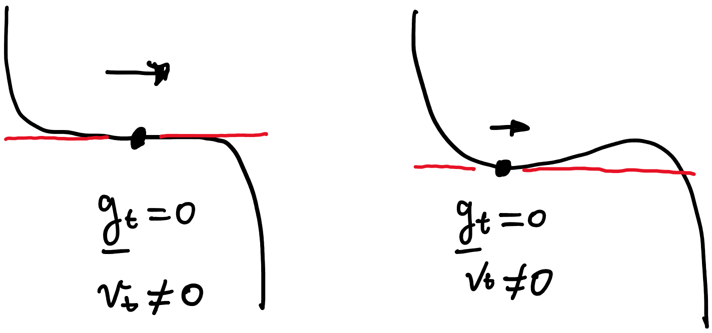
update: same as AdaGrad

Exponentially decaying moving average filter

act as momentum

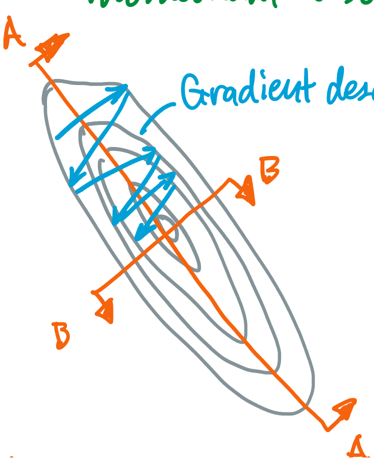
"Momentum": $\underline{v}_t = \alpha \underline{v}_{t-1} - \eta_t\, g_t$

update: $\underline{w}(t+1) = \underline{w}(t) + \underline{v}_t$



momentum helps solving non-convex surfaces,

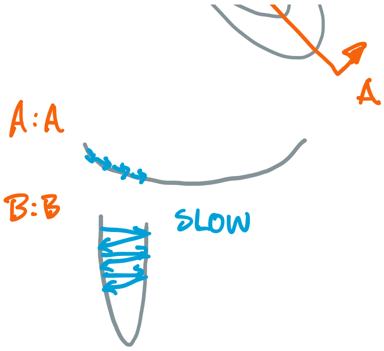
momentum also helps:



Gradient descent is ill-conditioned

Nesterov momentum:

$$\underline{v}_t = \alpha\, \underline{v}_{t-1} + \eta_t\, \nabla e_{n(t)}\left(\underline{w}(t) + \alpha\, v_{t-1}\right)$$



SDG with minibatch size 1

update $\underline{w}(t+1) = \underline{w}(t) + \underline{v}(t)$