Optimizations on Gradient Descent

23 September 2024 21:36

Momentum based GD → $w_{t+1} = w_t - \eta \nabla w_t$

but takes a lot of time to navigate a simple slope if I'm moving in one dirⁿ w/ many small steps I can take a big step.

$$\text{update}_t = r \cdot \text{update}_{t-1} + \eta \nabla w_t$$
$$w_{t+1} = w_t - \text{update}_t$$

} recursive too!

dude is capturing my history!

$\text{update}_0 = 0$

$\text{update}_1 = r\,\text{update}_0 + \eta \nabla w_1 = \eta \nabla w_1$

$\text{update}_2 = r\,\text{update}_1 + \eta \nabla w_2 = r\eta \nabla w_1 + \eta \nabla w_2$

$\text{update}_t = r \cdot \text{update}_{t-1} + \eta \nabla w_t$ → exponentially weighted avg.
→ as $t \uparrow$, 1st term ↓ weightage
→ $r < 1$ too.

- In regions w/ gentle slope — momentum based GD can take longer steps due to momentum.
- Sometimes moving fast = overshooting ∴
  - → momentum based GD may oscillate in & out of minimum valley.
    - → better than vanilla GD.
    - ↓ lots of u-turns before convergence.

Nesterov Accelerated Gradient: (NAG)

↓

"look before you leap." Before: $\text{update}_t = r\,\text{update}_{t-1} + \eta \nabla w_t$

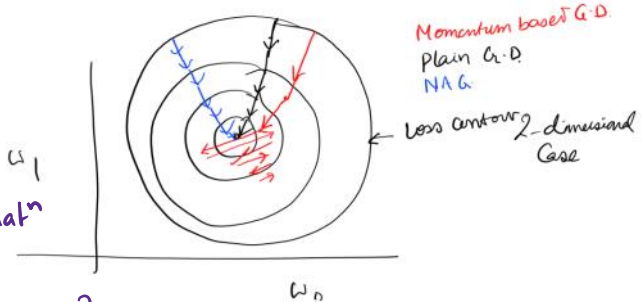
$r\,\text{update}_{t-1}$: move by this much; $\eta \nabla w_t$: and then some more.

But what if we calculated gradient @ partially updated value of w?

→ $w_t - r\,\text{update}_{t-1}$] look ahead

$$w_{\text{look\_ahead}} = w_t - r \cdot \text{update}_{t-1}$$
$$\text{update}_t = r\,\text{update}_{t-1} + \eta \nabla w_{\text{look}}$$
$$w_{t+1} = w_t - \text{update}_t$$

- dude corrects momentum GD's oscillations
- here, oscillations smaller
- chances of overshooting minima smaller.
- update @ current step = avg. of grad + prev step's estimatⁿ



Here's where it kinda sucks:

- treat w, b equally but loss surface never symmetrical
- slope high in one dimⁿ, low in another.

} Dimⁿ w/ large grad = small update vice versa.

Adjusting Learning Rate:

Gradient Descent	Stochastic G.D	Minibatch G.D.
→ goes over entire data before doing one update.	update after data point	update after a batch

Adjusting Learning Rate

Gradient Descent	Stochastic G.D	Minibatch G.D.
→ goes over entire data before doing one update. → True Gradient of Loss ↓ No approximation ↓ Theoretical guarantee holds ↓ very very slow <Extreme>	Update after data point <Extreme>	Update after a batch <in-between>

1 epoch = one pass over entire dataset
1 step = one update of the parameters $(\vec{w}, \vec{b})$
N = no of data points
B = mini batch size

Algorithm	# steps in 1 epoch
Vanilla G.D	1
Stochastic GD	N
Batch GD	N/B

Heuristics based learning rate optimizations:

① Tune learning rate:
- try diff values on log scale for η
- run a few epochs and find which best
- do finer search.

} linear search.

② Step decay:
- Halve η after every 5 epochs
- Halve η after epoch if validation error > prev epoch.

③ Exponential decay:
- $\eta = \eta_0 e^{-kt}$ t = step no, k, η_0 = hyperparams.

④ $\frac{1}{t}$ decay: $\eta = \frac{\eta_0}{1+kt}$.

Adagrad — gradient update integrating adaptive learning rate.

accumulated squares of gradients of each weight (cumulative)

$V_t = \nabla w_0^2 + \nabla w_1^2 + \dots \nabla w_t^2$ → used for normalization.

$$V_t = V_{t-1} + (\nabla w_t)^2$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{V_t + \epsilon}} \nabla w_t$$

Issues!
- decays η aggressively (denom ↑↑)
- freq. param = slow update

↓

fix by decaying denom!

↓

$V_t = \beta V_{t-1} + (1-\beta)(\nabla w_t)^2$

$$V_t = \beta V_{t-1} + (1-\beta)(\nabla W_t)^2$$

W_{t+1} = same.

ADAM — gradient descent (momentum) moving avg. + RMS prop moving avg. of sq. grad. → 2 decay hyperparams β_1, β_2

Now, the final update Rule (benefit of momentum approach; benefit of RMSprop approach; used in this eqn.)

$$W_{t+1} = W_t - \eta \frac{\bar{v}_t}{\sqrt{\bar{s}_t + \epsilon}} \nabla_{\bar{W}} \alpha\left(\bar{W}_t + \bar{b}\right)$$

Bias Correction in ADAM

Sum of weights of past values of state vectors $\bar{v}_t$, $\bar{s}_t$ will tend to ∞ if $t \to \infty$. This will happen only at very very large value of t.

So, for smaller values of t, we need to correct $\bar{s}_t$ & $\bar{v}_t$.

$$\hat{v}_t = \frac{\bar{v}_t}{(1-\beta_1^t)} \qquad \hat{s}_t = \frac{\bar{s}_t}{(1-\beta_2^t)}$$

works out when t is small!!