23-11-2021 Tuesday

Week #13 Lecture #16

what will be μ & σ in testing time?

we will save μ & σ for all trains set & use it in test time. we will compute μ & σ for all neurons

If we are not sending test images in batches.

We either send whole train set in NN & compute μ & σ for all neurons or send a major portion of Train set in NN & compute μ & σ for all neurons & use it in test time.

Optimizer:-

① Gradient Descent / Vanilla Descent:-

$w = w - \alpha dw$ $\quad dw = \frac{dL}{dw}$

Problem.

a) Same α (learning rate) value for all gradients.

Some dw are large

Some dw are small

So this leads to quick learning of first dw's & not for small dw's

② AdaGrad:- different α value for each dw.

$$\alpha_i = \frac{\alpha}{\sqrt{S}} \alpha_i \quad ; \alpha = 0.0001$$ usually in initial

$S_0 t = dw_1^2$ for w_1 for 32 batch input

So we get 32 dw we take mean of these 32 dw & this is dw_1

for next 32 images batch we again have 32 dw Now we take mean of these dw's Then we have dw_1 again

Now we add it to S

$$S_1 t = dw_1^2$$

whose learning rate was bigger, it will get smaller,

& those learning rate is smaller, it will get bigger.

Problem:

$\sqrt{S}$ term gets bigger & α gets smaller & our learning gets slow.

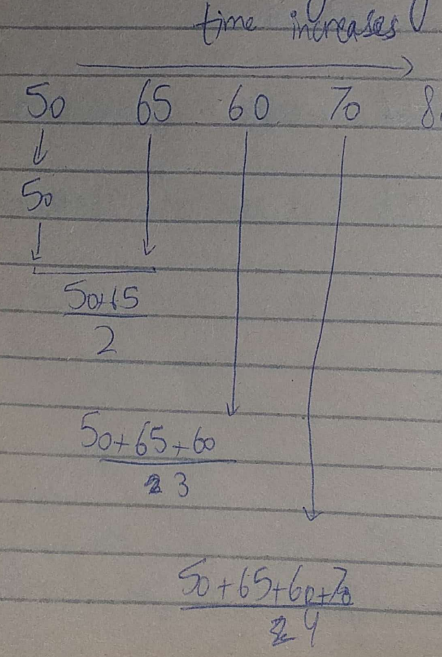
③ RMSProp:-

Instead of dividing by $\sqrt{S}$, we take avg of gradients & divide by that

$$\alpha_i = \frac{\alpha}{\sqrt{A}}$$

A = Cumulative Moving Avg

e.g



Problem:

New entry/data point isn't given much weightage in our avg.

So new derivatives aren't given much weightage

Solution:

Simple Moving Avg

50 65 60 70 80

$\frac{50+65+60}{3}$ then

$\frac{65+60+70}{3}$ Then $\frac{60+70+80}{3}$

window size = 3

$A = \frac{dW_1^{2[t_0]} + dW_1^{2[t_1]} + dW_1^{2[t_2]}}{3}$

t_0 = batch 1

t_1 = batch 2

t_2 = batch 3

Now if batch 4 comes.

$A = \frac{dW_1^{2[t_1]} + dW_1^{2[t_2]} + dW_1^{2[t_3]}}{3}$

Now taking Avg of last 3 derivatives of weights.

In RMS Prop, we use Decaying Moving Average bcz Problem in simple Moving Avg is that we are ignoring previous history of the derivatives

Decaying Moving Avg.

$$A_{t+1} = \beta A_{t-1} + (1-\beta) dw_1^2$$

$\beta = 0.9$ usually

A_{t-1} = Previous Avg

initially it could be 0 or $A_0 = dw_1^2$ [t=0]

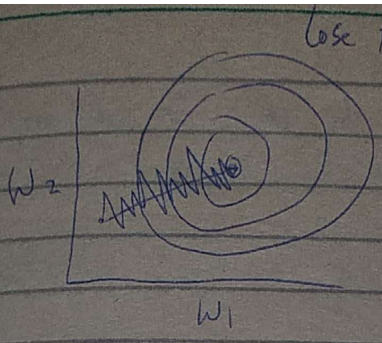
$$A_3 = \beta A_2 + (1-\beta) x_3$$

$$A_3 = \beta(\beta A_1 + (1-\beta) x_2) + (1-\beta) x_3$$

$$A_3 = \beta(\beta(\beta A_0 + (1-\beta) x_1) + (1-\beta) x_2) + (1-\beta) x_3$$

So historical Avgs are multiplied by β^n then β < 1 so β^n becomes small & historical Avgs get less weights.

④ Gradients with Momentum:-



Ideally we need to learn our weights in linear form in a straight line but our actual weights are updated in zig zag pattern

Oscillations Reason

Coz we are taking derivative of one weight we are assuming that others weights are constant and only one weight is updated but in reality other weights are also updated at same time so our learning of weights is not right.

2nd Problem.

We update weights on batches. So one batch might have different gradients than others so we have oscillations in our lose function.

Goal:
oscillations should be less in our improving of loss function.

Solution:-

$w_i = w_i - \alpha(\text{change})$

$w_i = w_i - \alpha(c)$ $\quad c = \text{change}$

$C_{t+1} = \beta C_t + (1-\beta)\, dw_i$

↙

by doing this, we get to know in what direction batches Avg are going so we update weights according

Initially, $C_0 = 0$ or $C_0 = dw_i^{[t_0]}$

Problem: usually we use 0.

In this, α value is constant.

⑤ ADAM:-

It has adaptive gradients & also lower we oscillations in this

$$w_i = w_i - \alpha_i C_t$$

$$C_t = \beta_1 C_{t-1} + (1-\beta_1)\, dw_i^{[t]}$$

$$\alpha_i = \frac{\alpha}{\sqrt{A_i}}, \quad A_i = \beta_2 A_{i-1} + (1-\beta_2)\, dw_i^2$$

C_t = Force $\quad$ β_1 is called Momentum

Convolution:-

We move a filter across the image & do some process, then at the end we get a filtered image.

e.g.

100	100	100
100	0	100
100	100	100

Image Part

1	1	1
1	1	1
1	1	1

Filter

We do element wise multiplication so we get

100×1	100×1	100×1
100×1	0×1	100×1
100×1	100×1	100×1

→

100	100	100
100	0	100
100	100	100

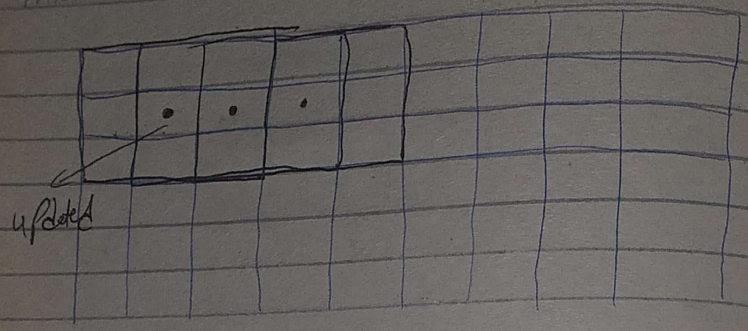
& we Sum up all the values in that result & also take Sum of all values in filter & divide Sum of result by Sum of filter

$= \frac{800}{9}$

$= 88.889$ we update 0 in original Image part & write this result.

100	100	100
100	88.89	100
100	100	100

This as a result reduces noises in the image.



we move Filter & process the image