Activations function in depth 1

→ In the last class we discuss, different activation function and talked about where to use which activation function.

I gave you kind of some general rules

→ Today we will talk about optimisers

(1) we ~~alrea~~ have already discuss about 1 optimiser gradient descent.

→ write the formula for gradient descent

$$w_{new} = w_{old} - \eta \frac{dL}{dw_{old}}$$

Gradient descent is most the most simplest optimisers. and even gradient descent 3 types.

(1) Batch gradient descent
(2) Mini Batch gradient descent
(3) Stochastic gradient descent

-> Thankfully, the formula for all this gradient descent doesn't change

$$w_{new} = w_{old} - \eta \frac{dL}{dw_{old}}$$

(1) Batch gradient descent -> So, what batch gradient descent does is, it calculates the differentiation that is $\frac{dL}{dw}$ for all the data points.

Let say we have 10000, data points, so, it will calculate differentiation for 10000 points.

-> once it has calculated the differentiation for all the 10000 data points it averages it out and update the weight.

Q -> So, what is the problem with batch gradient descent.

-> Since it has 10000 data points and it calculate the gradient for all this 10000 data points it becomes kind of very slow.

→ See, we have to update the weight for every epoch. and for each epoch: if we calculate the gradient descent for 10000 data points, this becomes very very slow.

→ Now, i have just taken 10000 data points, but some time in training dataset we have million, billons. in that case it will take a very long time just to update the weight. So, we do not want that

Hence, we have mini Batch gradient descent, Now as the Name suggest it will not used entire data point. So

MB. GD → $\frac{dL}{dw}$

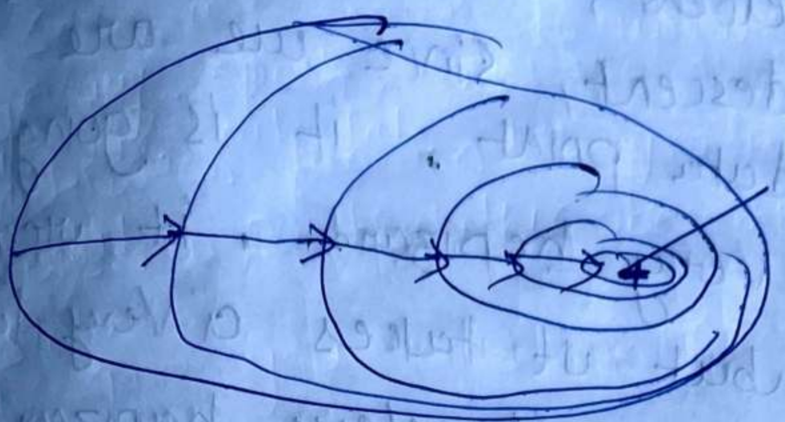
So, MB GD will do, from 10000 data point it will randomly select the 100 data points, it will calculate the differentiation of weights of this 100 data point and it will update the weights again ~~it~~ itself has some problem we we'll talk about later,

Then we have Sochastic Gradient descent

SGD → ~~[illegible]~~ in SGD they randomly select 1 data points and update the weights based on that.

So, the differentiation is calculated for 1 data points and weight are change based on the differentiation of that 1 data points.

Batch Gradient descent →



Here you can this indicates a loss function going inside towards the mining.

so, this is Batch gradient descent since we average out for all the data points it is verry very smooth.

So, it goes very smoothly, straigthy towards the centre.

in MBGD and SGD since we are not using all the data points, the differentiation $\frac{dL}{dw}$ for mini batch will be quite equal to differentiation $\frac{dL}{dw}$ for batch gradient descent but it will not be the same

$$MB \ \frac{dL}{dw} \sim \frac{dL}{dw} \ BGD$$

and Same thing chayppen with Stochastic gradient descent, So in stochastic gradient descent, since we are using only 1 data point. it is going to be very very hepizand, it reaches a centre but it takes a very long time, and it very hapizard.

-> So, what can we do, So, that we dont have to use Some many data points. and it require so much of time. but at the ~~Same time,~~

-> So, we use Some thing known as Stochastic gradient descent or mininbatch gradient descent with momentum

-> So, new term is added is the momentum term,

by the way i forgot to mention but mini batch gradient descent is most used gradient descent among all this 3. I THINK that is pretty understandable because that line was not that happizard and it does not take this much of time.

→ So if you see in the graph. there are two figures, this SGD, and SGD with momentum,

SGD → Here you can see, this very hapizard.

SGD with momentum → Here this is hapizard but it's not as hapizard as one with ~~no~~ momentum

Q → So, what is the very big idea behind with momentum

→ So, what momentum says. ~~instead~~ ~~of~~ ~~using~~ ~~instead~~ ~~of~~ ~~using~~ when differentiating we are using $\frac{dL}{dw'}$. instead of using $\frac{dL}{dw}$ differentiation for the last data point. let us calculate

differentiation for all the ~~datas~~ points
other step as well.

→ So, for any 1 datapoint instead of calculating differentiation for that point let us, some how add this data ~~pos~~ points as well into that

So, that means it will average out the points and wont be that hapizard

→ So, let me first change the notation

$$= w_{new} = w_{old} - W \frac{dL}{dw_{old}}$$

instead of new an old let us just write iteration number

t^{th} iteration, any iteration before t, is $t-1$ iteration, the old will replace with (t-1) and anything before that will be (t-2)(t-3) and so, on

$$w_t = w_{t-1} - \eta \frac{dL}{dw_{t-1}}$$

the reason why i am writing this is, because it becomes easier to write instead of writing new and old all the time. let w just represent this with a iteration number.

and instead of writing $\frac{dL}{dw}$ all the time, let me just replace with g_t (that is gradient tag and t_{th} point)

So, new equation becomes something like this

$$w_t = w_{t-1} - \eta g_t$$

→ Now, just lets get back to the formula. So, what happens is, the formula remains the same, we just have a new term. So, that means

$$w_t = w_{t-1} - v_t$$

$$\hookrightarrow v_t = \gamma v_{t-1} + \eta g_t,$$

where,

$$v_1 = \eta g_1$$

→ I know this is little complex, but
→ expand this

$$w_t = w_{t-1} - (\gamma v_{t-1} + \eta g_t)$$ with the formula ag so this will bri all h

$$= w_{t-1} - \eta g_t - \gamma v_{t-1}$$ if

we replace v_{t-1}:

$$- \gamma(v_{t-1} + \eta g_{t-1})$$

and this goes on an on. Honestly, and this comes out something like this

$$= w_{t-1} - \eta g_t - \gamma \eta g_{t-1} - \gamma^2 \eta g_{t-}$$

~~$w_t = w_{t-1} - (\gamma v_{t-1} + \eta g_{t-})$~~

$$w_t = w_{t-1} - \eta g_t - \gamma \eta g_{t-1} - \gamma^2 \eta g_{t-2}$$

$$- \gamma^3 g_{t-3} - \gamma^4 g_{t-4} \ldots$$

$$- \gamma^T g_1$$

-> So, what happens is. the γ (gamma) is always [0 - 1]

let assume gamma = 0.1

weight = 1

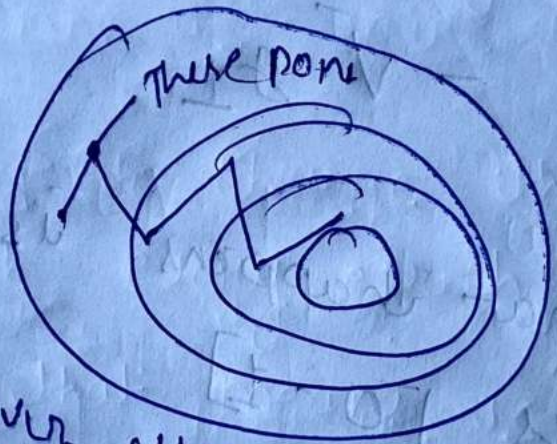
$$w_t = w_{t-1} - 1 \cdot \eta g_t - (0.1)^2 \eta g_{t-1}$$

$$- (0.01)^2 \eta g_{t-2}$$

Not the differentiation

-> at this t^{th} iteration the weight is one, for the iteration before that we have the weight of 0.1. So before that we have square $(0.01)^2$ and this goes on.

→ So, you can understand pattern, ~~wh~~ whe
happening is, you give the most
weigtage to the point ~~w~~ for which
it is being calculated but you
also give some weidage here to
there points.



→ However the weigtage keeps on
decreasing. So for the t^{th} iteration
you have a weidage of 1
for $t-1$ iteration you have weitage
of 0.1 . for gamma = 0.1

= for $t-2$ iteration you have
weitage $t-2 = 0.01$

and this goes on and on.

→ So, what happens is, since you are giving weitage to the previous point it does not becomes this happizerd

it kind averages out depending upon the position and it is happizerd yes, but it less happizerd now.

Momentum → momentum in short its basically means giving weitage to the previous calculated gradient descent ; so that the next gradient descent as hapizard as the previous one.

---

* Some other optimisers

① Adagrad
if you remember the equation of gradient descent is

$$w_t = w_{t-1} - \eta \, d\,gt$$

~~the~~