**Batch SGD with momentum**

As we know sgd updates are noisy. So to avoid this we use denoise which is shown below.

Suppose we have data a1 at t1, a2 at t2, a3 at t3 and so on.

So at t1, v1 = a1

At t2, v2 = gamma v1 + a2

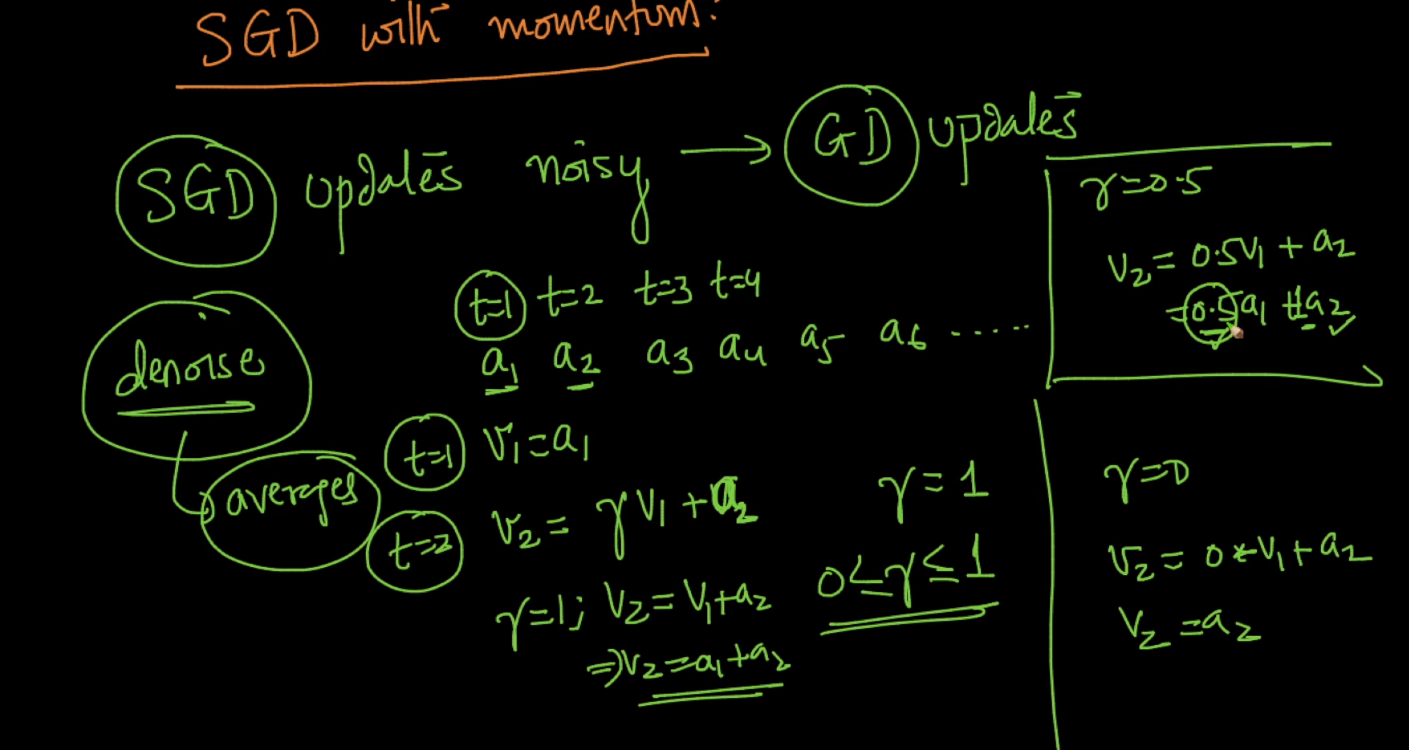
…

And gamma can be : 0 <= gamma <= 1

As shown below if we take gamma = 1 then v2 = v1 + a2 = a1+ a2

And if gamma = 0 then v2 = a2

Similarly for gamma = 0.5



We have to take initial v1 = a1 and we take gamma = 0.5

After that next data v2 = gamma\*v1 + a2 = 0.5 a1 + a2

Next data v3 = gamma \* v2 + a3

= gamma \*(gamma\*v1 + a2) + a3

= gamma2\*a1 + gamma\*a2 + a3

= 0.25\*a1 + 0.5\*a2 + a3

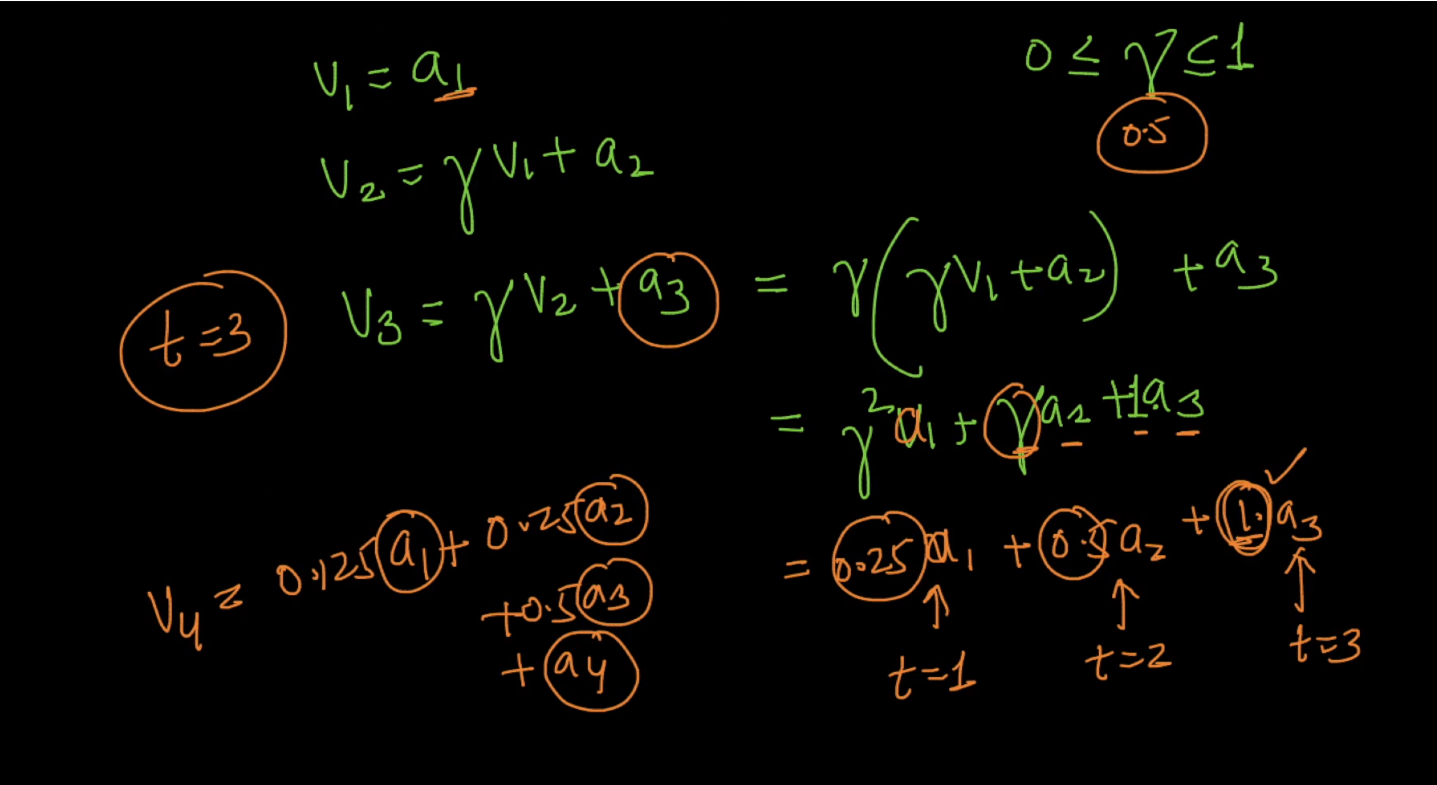
So here what we are doing is using all previous values to giving some momentum through exp weightage average or weightage sum average and by this we denoise the data by using previous data.

So In above example we can think as a weighted sum, giving more weightage of 1 the most recent point we have seen and less weightage to point we have seen in the past.

In v3 : a3 is most recent we give 1, a2 is less recent than v3 therefore we give less weightage to it than a3 i.e 0.5 and a1 is least recent therefore we give 0.25

In v2 : a2 is most recent we give 1, a1 we give 0.5

And similarly for v4



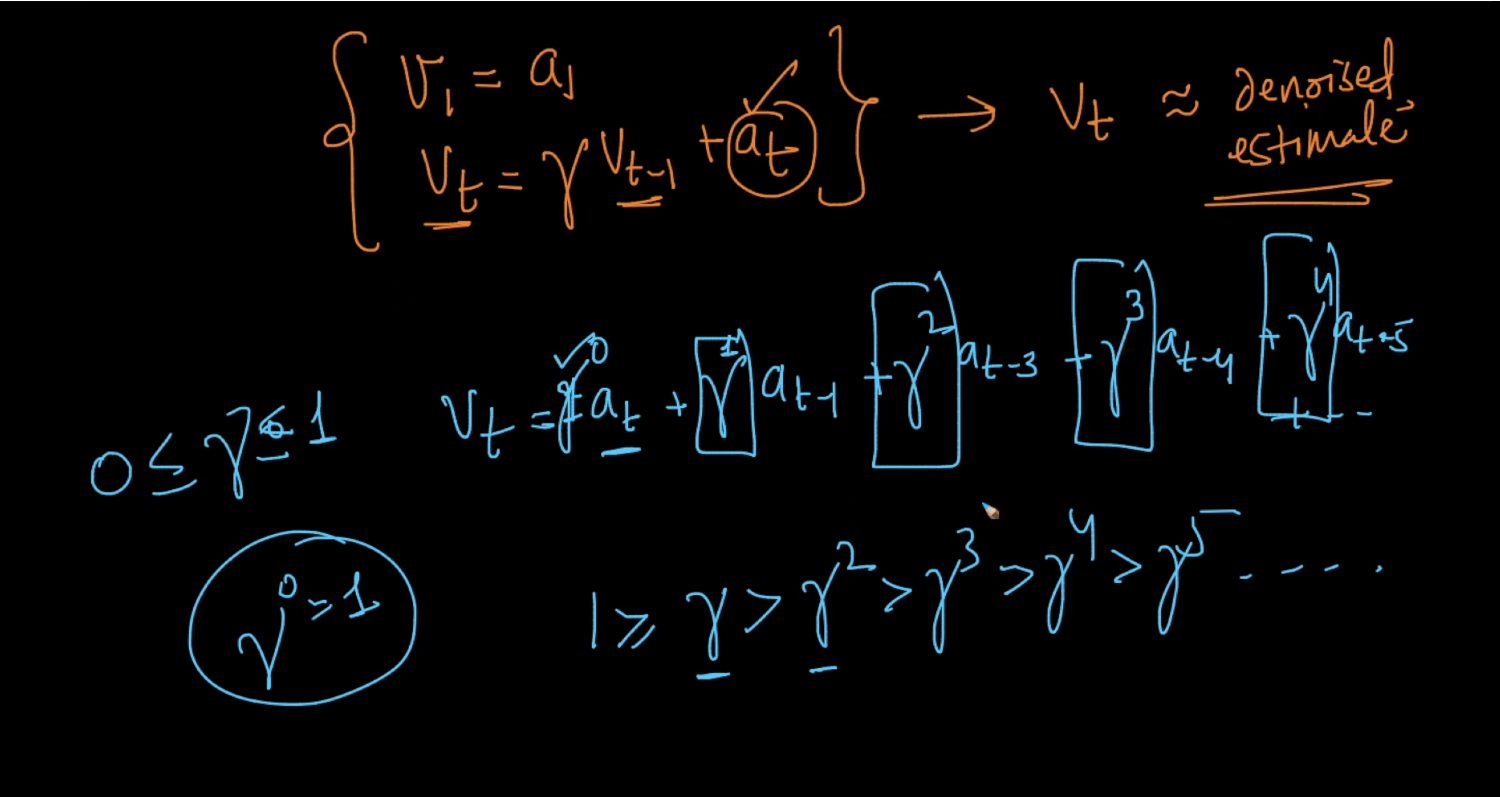
So the denoise eq. is given below it is recursive eq.

V1 = a1

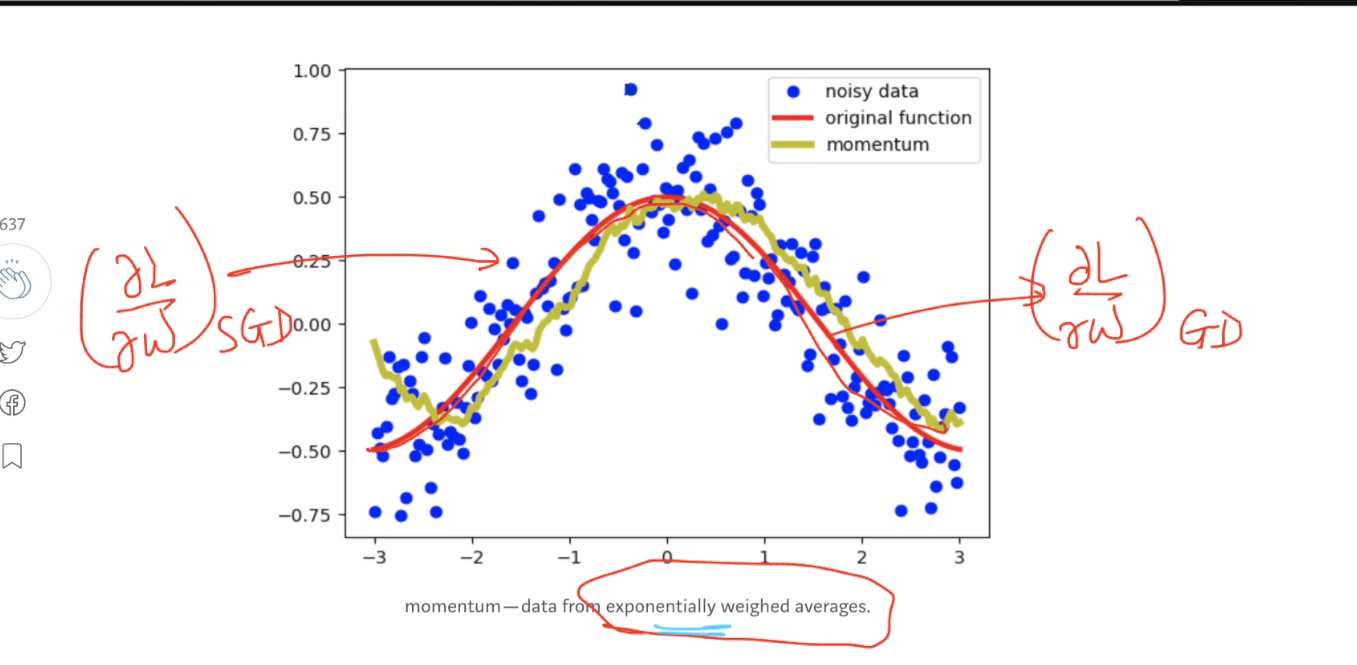
Vt = gamma Vt-1 + at

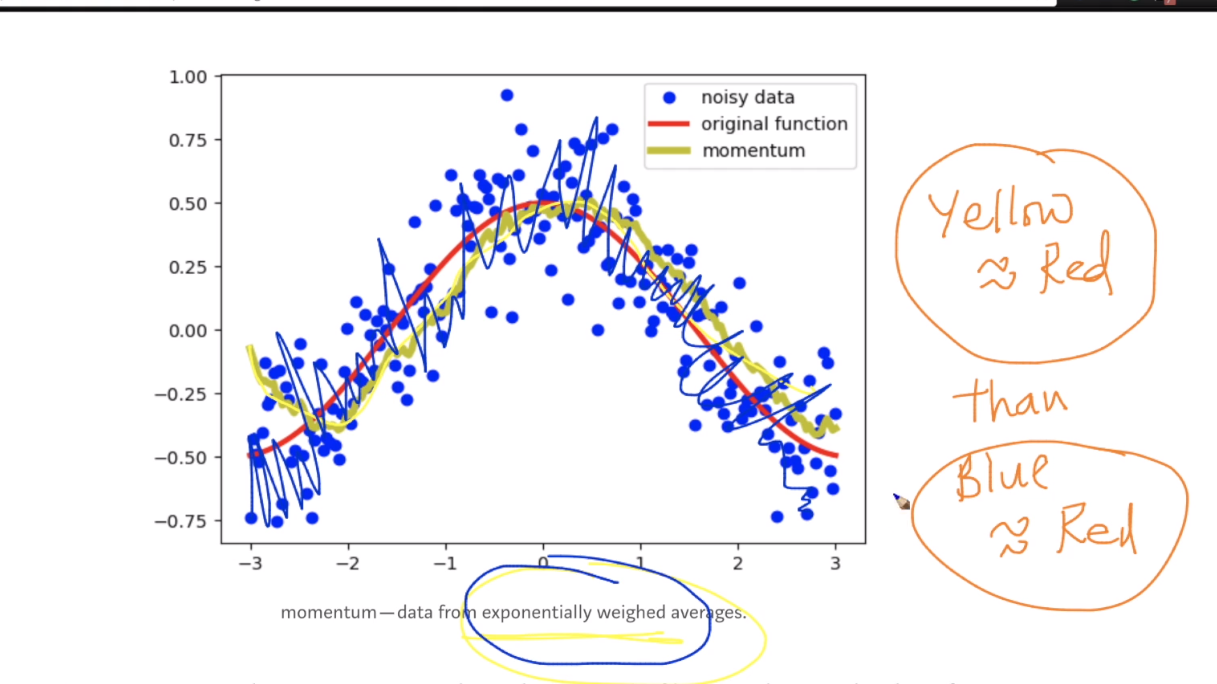
There Vt we got is approximate of denoised estimate.

So for denoised estimate we are using all previous values to giving some momentum exp. Weightage average as shown below.



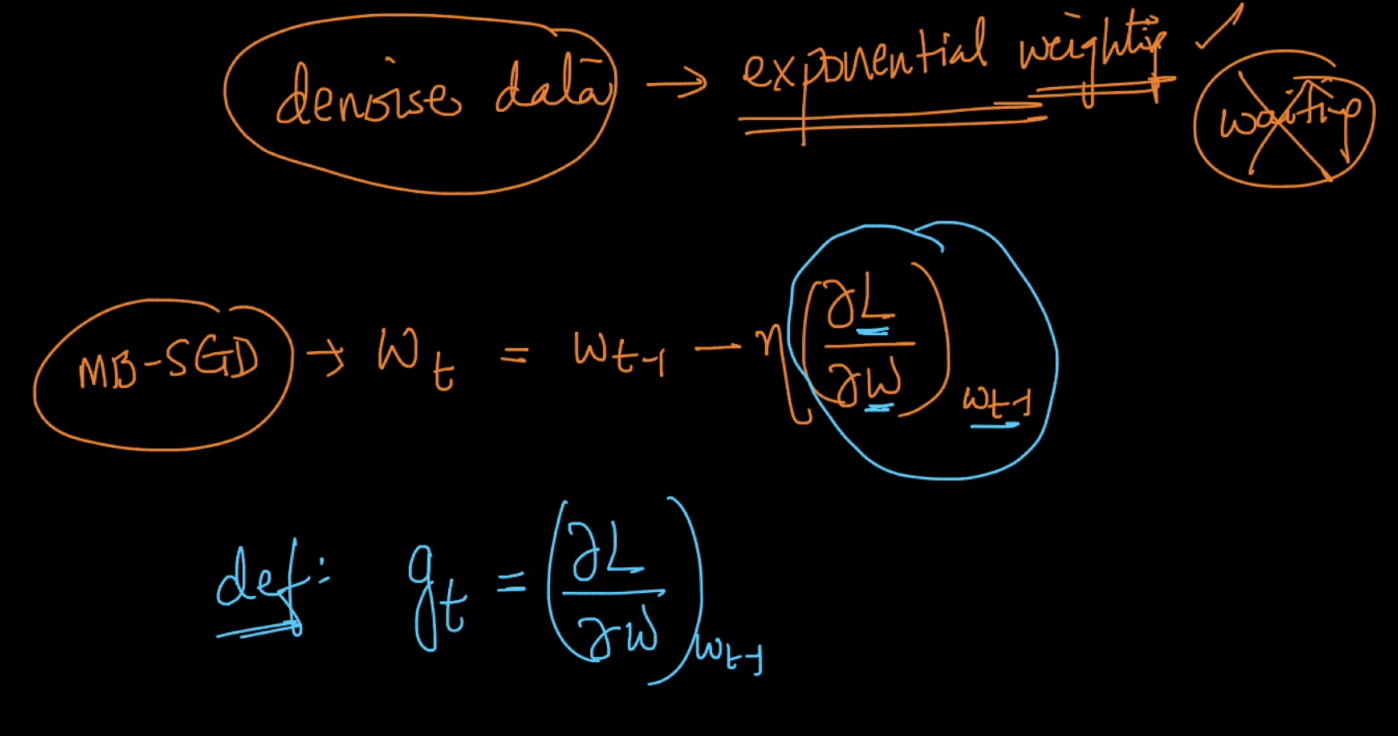
Below image shows red line it is for data given by gradient descent, blue points are for data given by mini-batch SGD which is noisy. And yellow show data from exponentially weighted average which is more approximate to red line.





We denoising data we use exponential weightage.

For mini batch SGD update eq. is given below and for gradient we use notation gt.



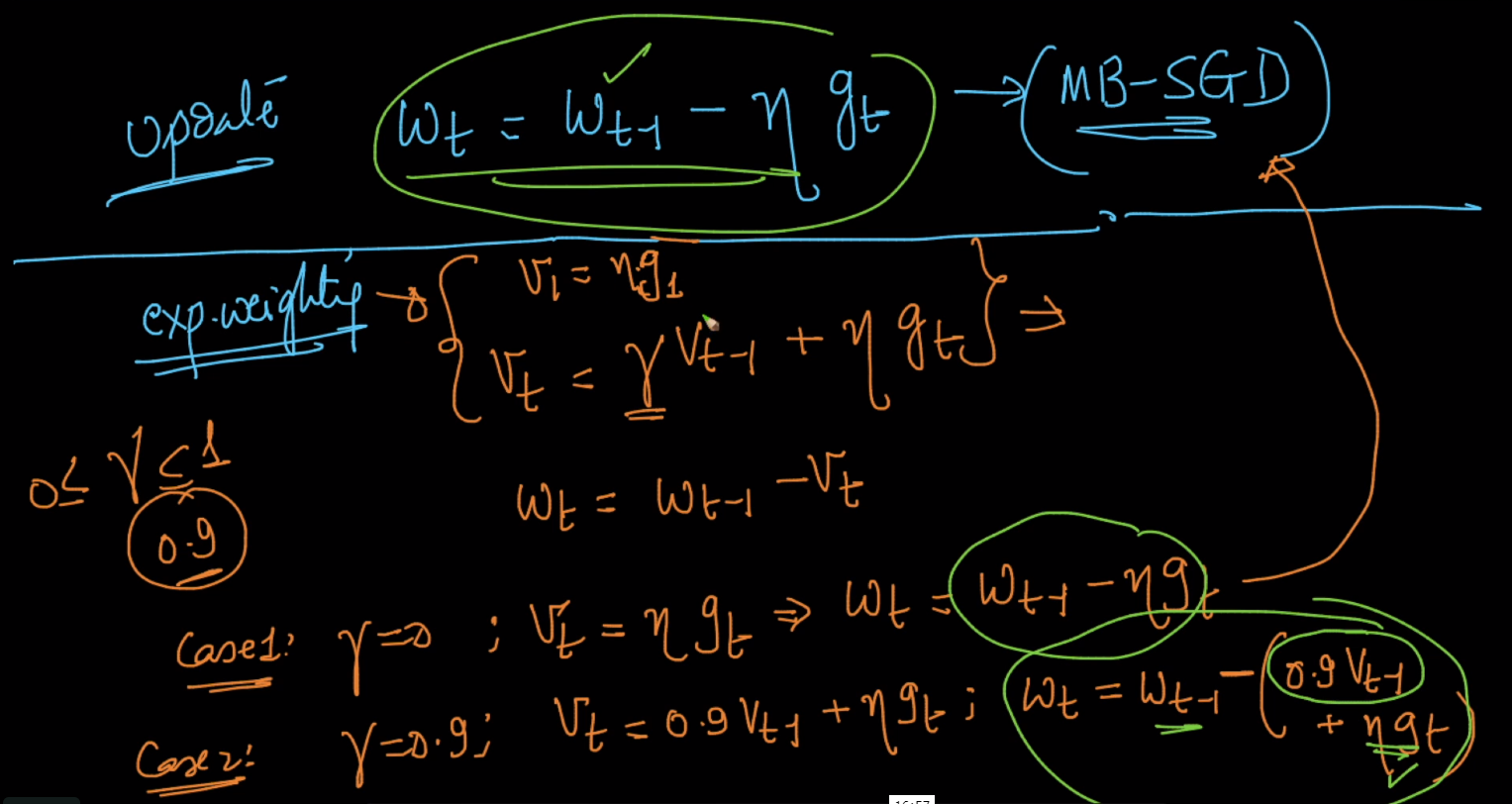
So for denoising of data we use exp. Weighting in update eq. of MB-SGD.

So below exp. weighting is same as above except we use eta\*g1 in place of a1 in v1

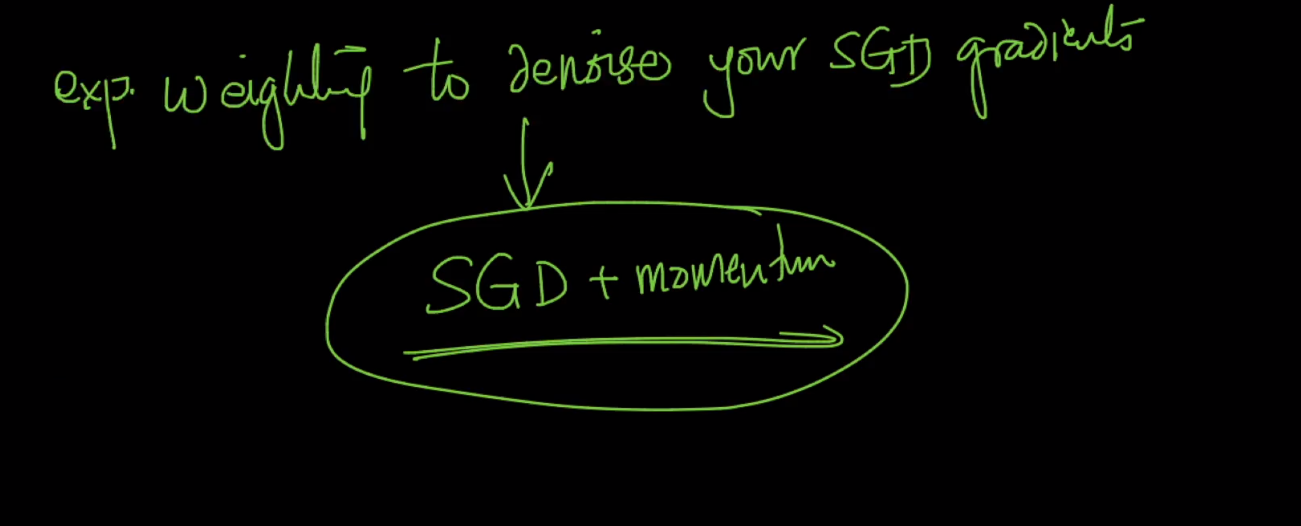
So new update eq. becomes Wt = Wt-1 – Vt

Case 1 : gamma = 0, Vt = eta\*gt => Wt = Wt-1 – eta\*g (it is same as normal MB-SGD update equation)

Generally we use gamma = 0.9 , by putting this in case2 we got eq. as shown below

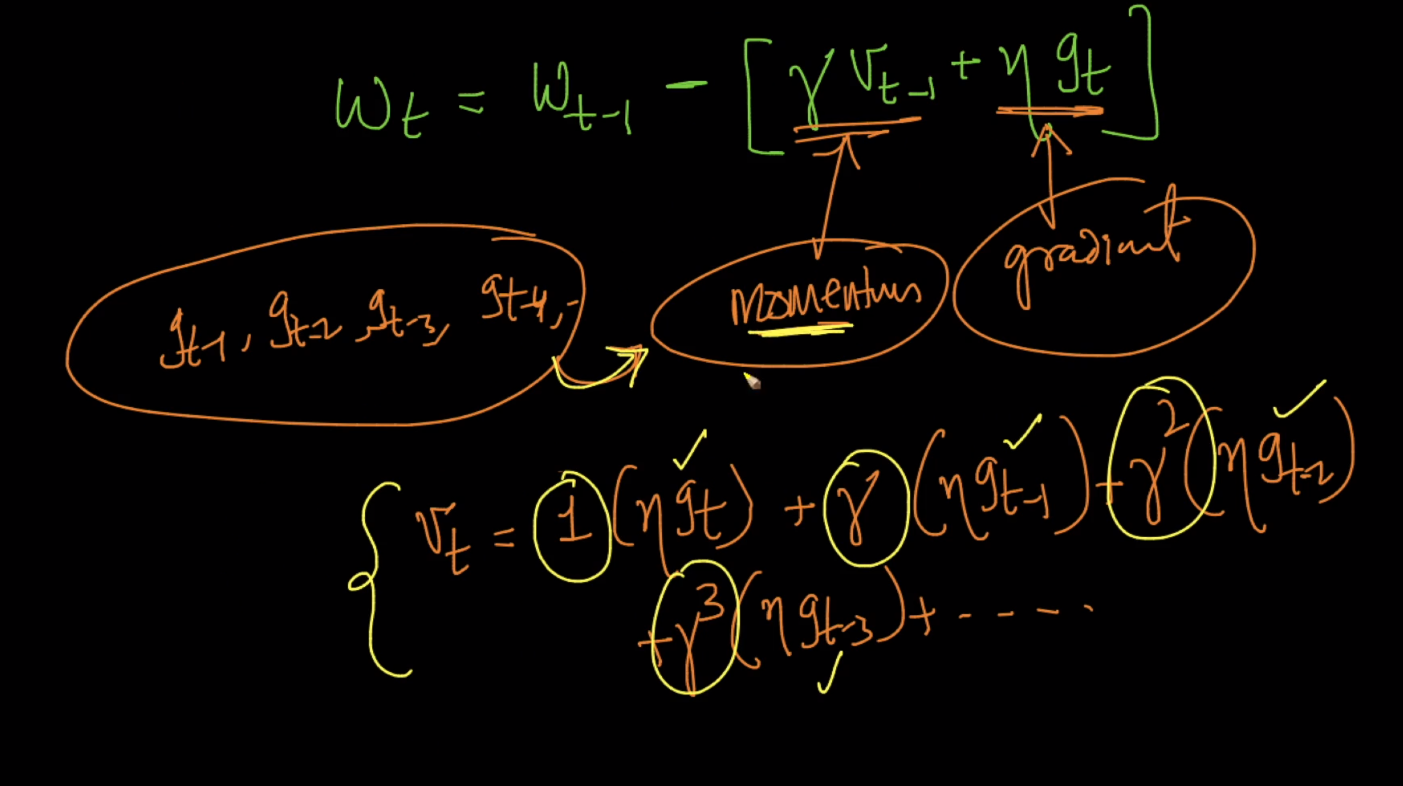


So exponential weighting to denoise our SGD gradients is SGD + momentum

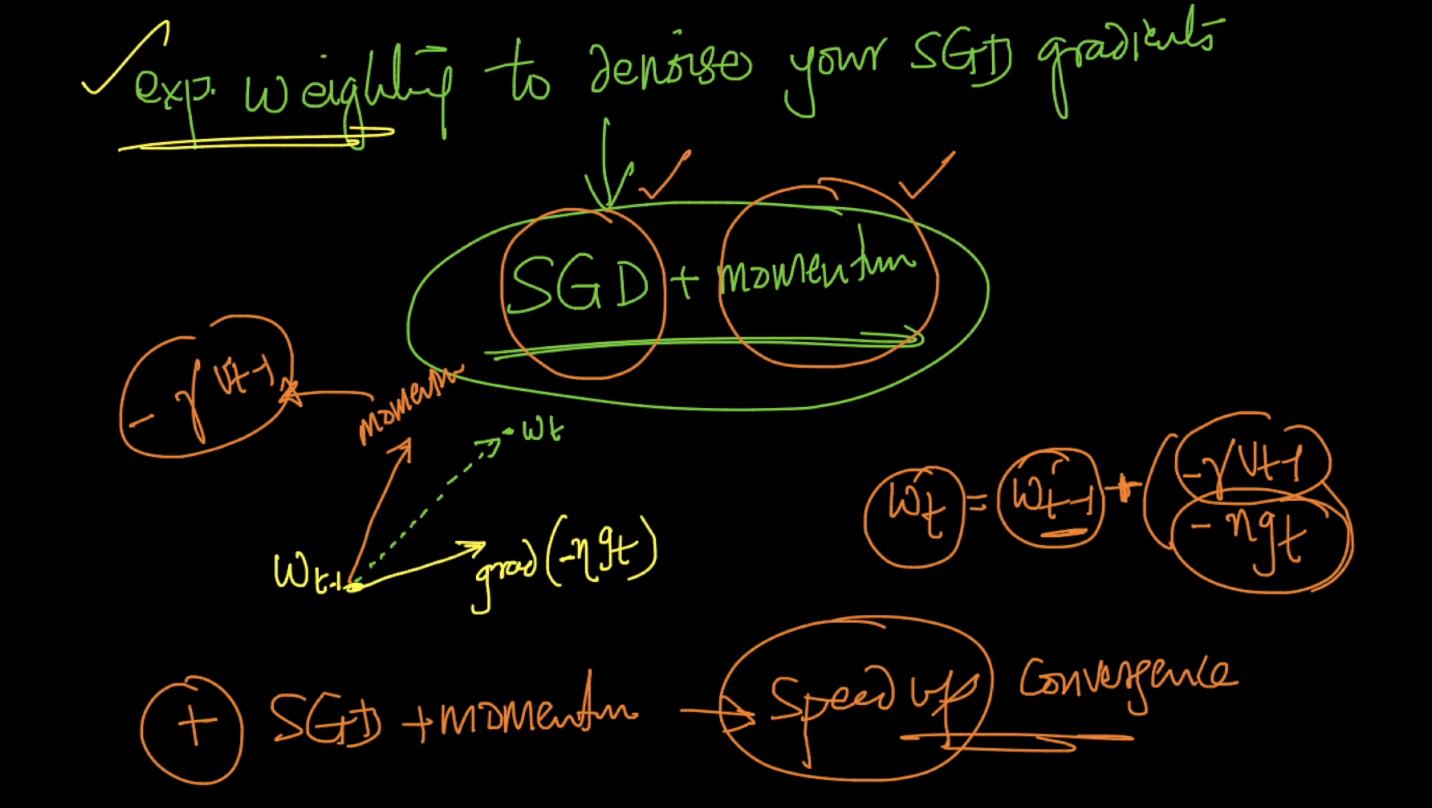


So in this update eq. momentum is gamma \* Vt-1 : in this we use previous gradient values to giving some momentum through exponential weighting to denoise data .

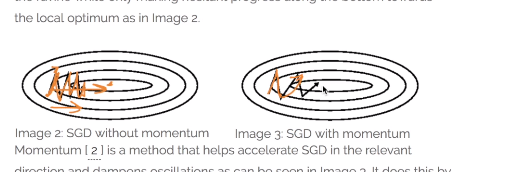
And we use same rule we give more weightage to recent data and less weightage to less recent data



So geometrically what happen is we for new update weight moves is gradient direction and if we add momentum then in momentum but by using SGD+momentum it moves in b/w by using exponential weighting sum in Wt direction and by this it decrease convergence time



First image shows convergence of MB-SGD it is more zig-zag and tooks lot of time and second is SDG with momentum so in this it uses previous knowledge and by this it knows that it moves in this direction only so it moves faster by less zig-zag pattern and converge faster



Link :

<https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d>

<http://ruder.io/optimizing-gradient-descent/>

<https://www.slideshare.net/SebastianRuder/optimization-for-deep-learning>

Comments :

