Gradient descent optimizers

- Stochastic gradient descent (**SGD**)

In SGD (with NO momentum) we update the weights and biases with the following formula:

$$b_t = b_{t-1} - \eta \frac{dL}{db_{t-1}}$$

where t is the current epoch and $t-1$ is the previous and L is the Loss function and η is the learning rate.

- Stochastic gradient descent (**SGD**) with **momentum**

Momentum is like rolling down a ball. It adds velocity. The weights are updated based on the last update of the weights multiplied by a **momentum term γ.** The momentum tells us how much importance we want to give to our previous updates.

Let's first discuss what is the **Exponential weighted average** . Let's assume we have a table

t1	t2	t3	t4	t5
a1	a2	a3	a4	a5

and a parameter γ (momentum) and calculate:

$$v_1 = a_1$$

$$v_2 = \gamma v_1 + (1 - \gamma)\, a_2$$
$$v_3 = \gamma v_2 + (1 - \gamma)\, a_3$$

and so on. In **SGD** is applied the same concept with respect to weights.

$$w_t = w_{t-1} - \eta\, v_{dw}$$

$$b_t = b_{t-1} - \eta \, v_{db}$$

Now let's calculate **Vdb** and **Vdw. The initial values of Vdb and Vdw are zeros.**

$$v_{dw_t} = \gamma \, v_{dw_{t-1}} + (1-\gamma)\frac{dL}{dw_{t-1}}$$

And analogical

$$v_{db_t} = \gamma \, v_{db_{t-1}} + (1-\gamma)\frac{dL}{db_{t-1}}$$

- **AdaGrad**

AdaGrad stands for **Adaptive Gradient**. In SGD the learning rate is the same for every weight. In AdaGrad the main idea is that **the learning rate is changed every step.**

$$w_t = w_{t-1} - \eta_t^{'} \frac{dL}{dw_{t-1}}$$

$$\eta_t^{'} = \frac{\eta}{\sqrt{\alpha_t + \epsilon}}$$

Where η is the initial learning rate. ε is a hyperparameter, usually very small number like 1e-7, because at the beginning $\alpha = 0$. And α can be calculated as:

$$\alpha_t = \sum_{i=1}^{t}\left(\frac{dL}{dw_i}\right)^2$$

There is a **disadvantage** of this approach. After some steps α will be a **very big number** and the **learning rate will become extremely small**.

- **RMSProp & AdaDelta**

Here we have just 1 change. η' is still present. We are just trying to solve the **AdaGrad** problem with extremely low learning rate at later steps. So everything remains the same, we only change the way we calculate the new learning rate.

$$\eta_t' = \frac{\eta}{\sqrt{S_{dw} + \epsilon}}$$

We can look at Sdw as moving average and it is calculated by the formula:

$$S_{dw_t} = \beta \, S_{dw_{t-1}} + (1 - \beta)\left(\frac{dL}{dw_{t-1}}\right)^2$$

Again at the **beginning Sdw is zeros.** β is our momentum term and is usually a bigger number. For example: **0.95.** This way **Sdw** cannot become very very big, because we give priority to the last value of **Sdw.** The other part of the algorithm remains the same.

$$w_t = w_{t-1} - \eta' \frac{dL}{dw_{t-1}}$$

- **Adam Optimizer**

Adam stands for **Adaptive Moment Estimation. Adam** optimizer is a **combination** of **SGD with Momentum** and **RMSProp.** With **momentum** we achieve **smoothening** and with **RMSProp** we achieve **a change of learning rate at each step.**

When we talk about **SGD with Momentum** we initialize 2 variables **Vdw** and **Vdb.**
When we talk about **RMSProp** we initialize 2 variables **Sdw** and **Sdb.**
Initially all these values will be 0.

First we are going to calculate the **derivative of Loss with the respect of weights** and the **derivative of Loss with the respect of biases.** And then:

Momentum :

$$v_{dw} = \beta_1 v_{dw} + (1-\beta_1)\frac{dL}{dw},\ v_{db} = \beta_1 v_{db} + (1-\beta_1)\frac{dL}{db}$$

RMSProp :

$$S_{dw} = \beta_2 S_{dw} + (1-\beta_2)\left(\frac{dL}{dw}\right)^2,\ S_{db} = \beta_2 S_{db} + (1-\beta_2)\left(\frac{dL}{db}\right)^2$$

And then we can update the weights and biases with the formula:

$$w_t = w_{t-1} - \frac{\eta\, v_{dw}}{\sqrt{S_{dw} + \epsilon}}$$

$$b_t = b_{t-1} - \frac{\eta\, v_{db}}{\sqrt{S_{db} + \epsilon}}$$

Where η is the **initial learning rate**.

There is a modification of Adam optimizer, called **Bias Correction**, where the only difference is that Vdw and Vdb are modified like:

$$v_{dw}^{correction} = \frac{v_{dw}}{1-\beta_1^t},\ v_{db}^{correction} = \frac{v_{db}}{1-\beta_1^t}$$

Similarly :

$$S_{dw}^{correction} = \frac{S_{dw}}{1-\beta_2^t},\ S_{db}^{correction} = \frac{S_{db}}{1-\beta_2^t}$$

Which adds some correction to the update parameters.