University of Tehran
Faculty of New Sciences and Technologies Department
of Mechatronics

# Practice of Artificial Neural Networks

By:
**Omid Moradi**

Lecturer:
**Dr. Masoudnia**

Fall 2024

## 1) How to accelerate gradient descent optimization by momentum term and second-order term (adam optimizer)?

### Gradient Descent With Momentum

The problem with gradient descent is that the weight update at a moment (t) is governed by the learning rate and gradient at that moment only. It doesn't take into account the past steps taken while traversing the cost space.

Let's assume the initial weights of the network under consideration correspond to point A. With gradient descent, the Loss function decreases rapidly along the slope AB as the gradient along this slope is high. But as soon as it reaches point B the gradient becomes very low. The weight updates around B is very small. Even after many iterations, the cost moves very slowly before getting stuck at a point where the gradient eventually becomes zero.

In this case, ideally, cost should have moved to the global minima point C, but because the gradient disappears at point B, we are stuck with a sub-optimal solution.

To account for the momentum, we can use a moving average over the past gradients. In regions where the gradient is high like AB, weight updates will be large. Thus, in a way we are gathering momentum by taking a moving average over these gradients. But there is a problem with this method, it considers all the gradients over iterations with equal weightage. The gradient at t=0 has equal weightage as that of the gradient at current iteration t. We need to use some sort of weighted average of the past gradients such that the recent gradients are given more weightage.

This can be done by using an Exponential Moving Average(EMA). An exponential moving average is a moving average that assigns a greater weight on the most recent values.

The EMA for a series *Y* may be calculated recursively

$$s(t) = \begin{cases} Y(1) & t = 1 \\ \beta * s(t-1) + (1-\beta) * Y(t) & t > 1 \end{cases}$$

where

- The coefficient β represents the degree of weighting increase, a constant smoothing factor between 0 and 1. A lower β discounts older observations faster.
- *Y(t)* is the value at a period *t*.
- *S(t)* is the value of the EMA at any period *t*.

In our case of a sequence of gradients, the new weight update equation at iteration t becomes

$$v(t) = \beta * v(t-1) + (1-\beta) * \delta(t)$$

**Note**: In many texts, you might find (1-β) replaced with η the learning rate.

From above, we can deduce that higher β will accommodate more gradients from the past. Hence, generally, β is kept around 0.9 in most of the cases.

**Note**: The actual contribution of each gradient in the weight update will be further subjected to the learning rate.

When all the past gradients have the same sign, The summation term will become large and we will take large steps while updating the weights. Along the curve BC, even if the

learning rate is low, all the gradients along the curve will have the same direction(sign) thus increasing the momentum and accelerating the descent.

when some of the gradients have +ve sign whereas others have -ve, The summation term will become small and weight updates will be small. If the learning rate is high, the gradient at each iteration around the valley C will alter its sign between +ve and -ve and after few oscillations, the sum of past gradients will become small. Thus, making small updates in the weights from there on and damping the oscillations.

## How Does Adam Optimization Work?

So, that is all great, but how does this adaptive learning rate strategy work? Let's try to understand it through an example. Think of Adam as a father teaching his two kids, Chris and Sam, how to ride bikes. They have a problem, though: While Chris is afraid to pick up speed on his bike, Sam is more daring and pedals very fast. If Adam pushes both bikes at the same speed, Chris will lag because he's too slow, while Sam might crash going too fast!

So, Adam keeps noting the speed and acceleration of each kid and uses that information to adapt his strategy to push them. He notices that Chris has been going slow in general. So, he gently pushes Chris's bike harder to speed him up. Likewise, he sees Sam has been pedaling very quickly, so he lightly pushes Sam's bike to slow him down a bit. By adaptively adjusting the speed for each kid, he can train both of them at the right pace.

Chris picks up speed comfortably while Sam avoids accidents.

The Adam optimization works similarly. It customizes each parameter's learning rate based on its gradient history, and this adjustment helps the neural network learn efficiently as a whole.

Here is the simplified version of Adam:

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$$
$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2$$
$$\theta = \theta - (\alpha * m_t / \sqrt{(v_t + \varepsilon)})$$

Adam combines the two approaches through hyperparameters $\beta_1$ and $\beta_2$. We can see how the $\theta$ update looks similar to the RMSProp update, amplified with Momentum, so we get the best of both words.

In its final version, Adam also introduces a warm start. The t value varies from one to num_steps. It warm starts the algorithm when the start $m_{t-1}$ and $v_{t-1}$ values are zero.

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$$
$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2$$
$$\hat{m}_t = m_t / (1 - \beta_1^t)$$
$$\hat{v}_t = v_t / (1 - \beta_2^t)$$
$$\theta = \theta - (\alpha * \hat{m}_t / \sqrt{(\hat{v}_t + \varepsilon)})$$

The main hyperparameters in Adam are:

- $\alpha$—Step size for the optimization.

- $\beta_1$—Decay rate for momentum. The typical value is 0.9 as we want to have high weighting for the most recent gradients.
- $\beta_2$—Decay rate for squared gradients. The typical value is 0.999. This is because 0.999 captures the long-term memory of gradients. We want a stable estimate of variance.
- $\epsilon$— Small value to prevent division by zero. It is usually around 1e-8.

  Together, these parameters enable Adam to converge faster while remaining numerically stable. The defaults work well for most cases but can be tuned if needed.

## References


Gradient Descent With Momentum

Complete Guide to the Adam Optimization Algorithm