# Mr. Adam Who Are You?

Kuba Jerzmaowski

Thursday 20th June, 2024

## 1 Introduction

This here is an explanation of Adam I made an exercise in understanding. If you reference lesson 00, I mentioned that we will use Adam over SGD as it is better, and we will discuss Adam later. Now is later!

## 2 Issues with Standard SGD

### 2.1 Issue 1: Navigating Local Minima

Sometimes as we are adjusting our weights, we will be moving in one direction for a while, and then we get to a spot that tells us to move the weight back in the opposite direction (each parameter and weight can only move in two directions: up/down, right/left, +/- regardless of the vernacular you use). This change in direction can be for many reasons, but one such reason could be the presence of a hump as visualized in (Figure 1). In this case, we would like the gradient to get past the hump and not get stuck in the local minimum but rather reach the global minimum to the right. This is where the idea of momentum comes along. Given that we remember past gradients, we can use their past momentum to guide us in the current period. This is much akin to momentum in real life.

### 2.2 Issue 2: Adaptive Learning Rates

Now we know that learning rate is an important hyperparameter because if it's too low, we will learn very slowly and take forever to converge. However, if it's too big, then we will jump around our loss surface and overshoot our steps, leading to ineffective learning. That being said, if you're in a huge slowly downward sloping plane (Figure 2a), you would like to take a big step to accelerate the learning process. However, if you find yourself in a place where the curvature of the loss function is wild and changing a lot (Figure 2b), you want to take very small steps to ensure you're learning carefully and don't fall off any cliffs. When we walk around adjusting our parameters, we don't actually see the loss surface; we see the gradient and which direction the gradient points us. Then we can find the second derivative (or an approximation of it), which will tell us

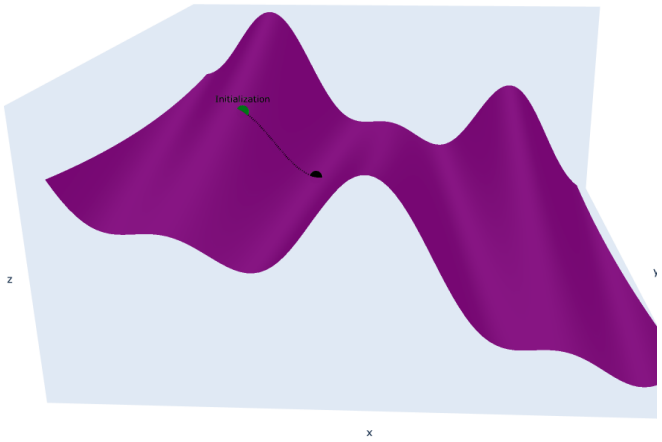
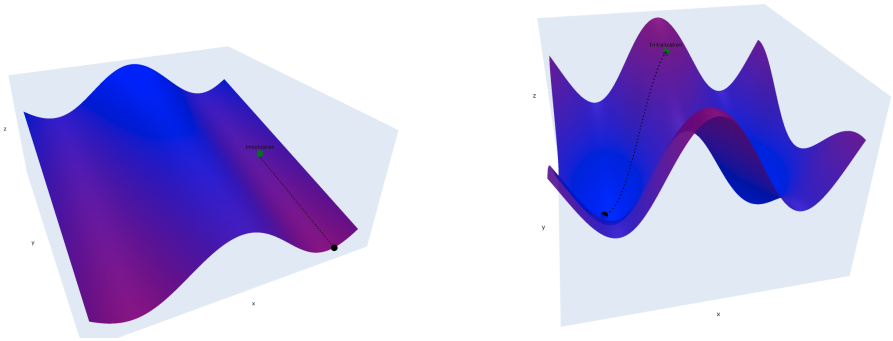



Figure 1: SGD getting stuck in local min

the rate of change of the rate of change, akin to the curvature of the spot we are in.



(a) Small curvature, small magnitude of change

(b) Large curvature, large magnitude of change

Figure 2: Comparison of curvatures and magnitudes of change

## 3 The Adam Algorithm

Given these two issues and the general mention of fixes that can be used, let's examine the Adam algorithm for PyTorch. (Algorithm 1, and I have slightly

removed some parts that correspond to other additions to the algorithm that we will not be discussing)

**Algorithm 1** Adam Algorithm

**Require:** $\alpha$ (learning rate), $\beta_1, \beta_2 \in [0, 1)$ (exponential decay rates for the moment estimates), $f(\theta)$ (objective function with parameters $\theta$)
Initialize $m_0 \leftarrow 0$ (initial 1st moment vector)
Initialize $v_0 \leftarrow 0$ (initial 2nd moment vector)
Initialize $t \leftarrow 0$ (initial time step)
**for** $t = 1$ to .... **do**
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (get the gradient at time step $t$)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
$\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$
$\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$ (update parameters)
**end for**

### 3.1 Step 1: Gradient Assignment

We assign the gradient calculated by backpropagation in time period $t$ to $g_t$ .

### 3.2 Step 2: First Moment Calculation

We assign $m_t$, which is the first moment, to the exponentially weighted average of the gradients. So our $\beta_1$, a hyperparameter, is the weight "strength of the momentum" to put on average past gradients. Often it's 0.9, so then it would look something like this:

$$m_t = .9 * m_{t-1} + .1 * g_t \tag{1}$$

Now, ignoring the first case, $m_{t-1}$ is just the average of the past gradients, so we weight that with 0.9 and our gradient calculated in backpropagation with the weight 0.1, then we sum to get the correct gradient.

This $m_t$ is how we account for issue 1. Since we heavily weigh past gradients, we will continue in that direction even if the current gradient tells us to go in the opposite direction. This will help us in the case of something like Figure 1. Now, if we continue to accumulate a new gradient that tell us to go in the opposite direction, then we will start to go in the opposite direction, which could lead to a problem like in Figure 3.

### 3.3 Step 3: Second Moment Calculation

Then $v_t$ is identical in that we weight the past $v_t$ often with 0.999 and sum it with the weight of the element-wise squaring of the current gradient calculated

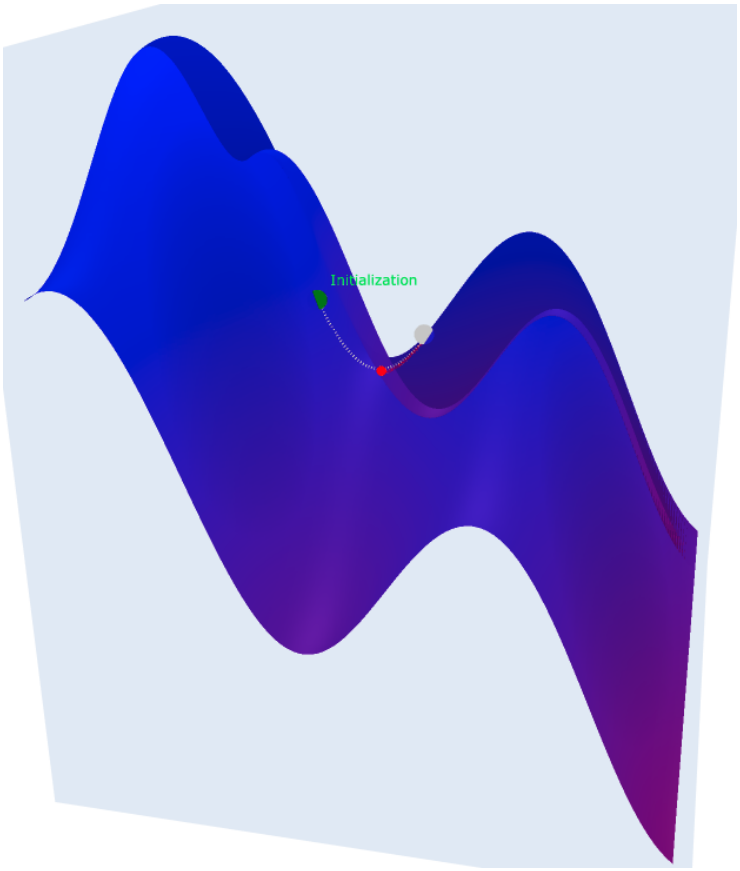



Figure 3: Unlucky initialization at green. Momentum will take you to white point. However not enough to get over hill and you fail at red point.

from the latest backpropagation in time period $t$.

But what is $v_t$ and what does it do for us? $v_t$ represents the second moment or the exponentially weighted average of the gradients squared. By squaring the gradient, we are capturing the variance or magnitude of the change. Now you can see how this starts to be a solution to issue number 2. If we know the magnitude of the change (the curvature of the surface), then we can use that to adjust our learning rate dynamically for each parameter individually.

### 3.4 Step 4: Bias Correction

The next two lines are doing the same thing for $m_t$ and $v_t$ respectively. This is the bias correction step. Both $\beta_1^t$ and $\beta_2^t$ approach 0 as $t \to \infty$ and $(1-\beta^t) \to 1$. This means that $\hat{m_t}$ and $\hat{v_t}$ are just what we calculated above since we are dividing by 1. So if we are just dividing by 1 whats the point of this step? This is important because in the first iteration, $m_t$ is initialized to zero, so the equation looks like this:

$$m_1 = 0.9 \cdot 0 + 0.1 \cdot g_1 = 0.1 g_1 \tag{2}$$

Which you can see results in the average of our past gradient being zero and biases the entire first moment toward zero. This would, in effect, severely decrease the learning rate and cause the learning in the beginning epochs to be slow. So if we do the unbiasing step in epoch one:

$$\hat{m_1} = \frac{m_1}{1-\beta_1^1} = \frac{0.1 g_1}{1-0.9} = g_1 \tag{3}$$

Now you can see how this helps our learning rate not be abysmal at the start, and the same goes for $\hat{v_t}$.

### 3.5 Step 5: Parameter Update

So now we get to the fun part: the update step. As always, our new params are equal to our old params minus the gradient step times the learning rate. But let's focus on what's the "gradient".

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{4}$$

So $\hat{m_t}$ is our first moment, meaning it represents the direction in which we want to go, accounting for momentum. The division in the update step addresses issue 2. We want to take small steps when the rate of change is changing a lot (lots of curvature) and bigger steps when it's not (less curvature). If we divide a number by a very big number, we will get a small number; if we divide a number by a small number, we will get a big number. So if the denominator corresponded to the magnitude of the change, we would be golden. Lucky for us that is what $\hat{v_t}$, the second moment, represents.

The epsilon term (where did this buddy come from?) is always some very small constant. If the magnitude of our change $\hat{v}_t$ is zero, we don't want to divide by zero, and with floating-point rounding in computers, we could run into other issues. So epsilon is there to keep us safe from those scenarios.

As for why we take the square root of $\hat{v}_t$: the second moment estimate $v_t$ can vary a lot. By taking the square root of it, we "normalize" it, preventing large fluctuations in our adaptive learning rate. If $v_t$ was 100 and we did not take the square root of it, then the step size we would take would be very small, but if we square root it, then it's 10, and that's a more reasonable step size.

# 4 Conclusion

Well that was Mr. Adam a pretty cool guy might I say. While Rick Blaine and Captain Louis did not build neural networks if they would I think they would have said something like this after learning about Adam. (note they would have to adjust Adam if they wanted too use weight )

> "Louis, I think this is the beginning of a beautifully optimized model with Adam" - Casablanca (1942)

**Last Modified:** Thursday 20$^{\text{th}}$ June, 2024