<https://cognizant.udemy.com/course/data-science-deep-learning-in-theano-tensorflow/learn/lecture/25807430#overview>

Suppose I have a set of data points X1X2 all the way up to x t.

The average of these data points is exactly what you expect.

Sum up all the x's and divide by the total number of data points, which is t.

Okay.

So pretty simple so far.

The next step is to consider what if I have so much data that I cannot fit it all into memory at once?

For example, the data lives in a file that's one terabyte in size.

Unfortunately, your computer is not capable of storing one terabyte of memory.

Or consider another scenario where you have a robot interacting with the world.

In this case, each data point arrives at the robot sensors at each instant in time.

I can't see any of the future data points, so my average can only consist of the past data points and

the current data point.

If I want the average of all the data points I have seen so far, let's assume that the current time

is T, then sure, I can add them all up and divide by T, but that seems wasteful.

You can see that as t increases the number of things I have to add together also increases.

Therefore, on each step my robot will have to do more and more computation in order to compute this

average.

We say that this computation is o of T because the number of steps to perform the computation is proportional

to T.

My claim is that if we have a robot streaming in data one point at a time, we can actually make this

computation o of one.

In other words, our computation is constant.

No matter how much data we've collected specifically, that we can compute the current average by somehow

making use of the previous computations we've done, and even more specifically, the previous average.

To understand how we can do this.

Let's manipulate the equation for the sample mean.

Firstly, notice that our notation for the mean at time t is x bar subscript t.

Let's begin by splitting our sum.

By taking out only the final data point X of T.

The next step is to multiply out the one over T term.

The next step is to replace the sum from little T equals one up to big T minus one.

Specifically we can replace this with the sample mean at time t minus one.

As you recall, the sample mean at time at t minus one is just the sum divided by t minus one.

Okay, so now what do we have?

We have exactly what we discussed previously.

We have an equation that allows us to compute the current average x bar T in constant time.

As you can see, it only involves summing two terms.

The previous average x bar t minus one and the current sample X of T.

The next step is to manipulate our expression.

Even more specifically, let's take the first constant T minus one over T and divide everything by T.

Then we get one minus one over T.

This is helpful because now the one over t term appears twice.

The next step is to ask the question What if we replace one over T with a constant.

Let's call it alpha.

In this case, we are changing the answer, but we are doing so in a way that might be helpful later

on when this was equal to one over t we saw that this gave us the regular sample mean.

As you can see, the function one over t decreases as t increases.

However, it turns out that this will weight all of the x's justrillionight so that every x we have

considered so far is weighted by one over T.

Note that we have just proven this and as you know it leads exactly to the sample mean.

Importantly, this means that if we replace one over t with a constant we will no longer have the sample

mean.

In fact, when we have a constant, this gives us the exponentially weighted moving average.

As an exercise, you might want to try manipulating the equation for constant alpha and put it back

into a summation so that it depends only on the X's and not any previous averages.

What you should see is that the weights decay exponentially as you go further and further back in time.

This is opposed to the regular average where all the weights are equal.

Basically, this means that for the exponentially weighted moving average data points which are more

recent are more highly weighted.

…

What is the sample mean?

You may recall from statistics that the sample mean is used to estimate expected values.

Now, what is the expected value of the square of a random variable?

Well, it is related to the variance.

Specifically, the variance is equal to the expected value of the random variable squared minus the

true mean squared.

However, when we just have the expected value of the square, we often call this the second moment.

In general, the expected value of X to the power n is called The Nth Moment.

In statistics, it's often the case that what we care about are the first and second moments.

There are some practical and theoretical reasons behind this, but they are outside the scope of this

lecture.

Intuitively, we know that what we often care about when it comes to something random is the mean and

the variance.

We know that the second moment is something like variance.

Similarly, the first moment is simply the mean that is the expected value of X to the one or simply

the expected value of x is the mean.

At this point, we can recognize how this relates to momentum and our RMS prop momentum essentially

estimates the first moment of the gradient using an exponential moving average.

Similarly, RMS prop essentially estimates the second moment of the gradient using an exponential moving

average.

Now, since we might want to tune these two values separately, we will give each of them a possibly

different value of beta.

We'll call the beta for the first moment, beta one, and we'll call the beta for the second moment

beta two.

The last step is to essentially put these two ideas together and that gives us Adam.

In order to make sense of how to put these two ideas together, let's remind ourselves what momentum

and RMS prop look like when they are apart.

For momentum, we subtract the learning rate and multiply by M.

For our RMS prop, we subtract the learning rate, multiply by the gradient and divide by the square

root of V plus some tiny value.

Thus for Adam, we simply combine these two together.

That is to say, we subtract the learning rate, multiply it by m and divide by the square root of V

plus the usual tiny value to avoid dividing by zero.

Now this is the essence of the Adam Optimizer.

But note that there is still one more step before we are done.

Since this lecture has been quite long so far and this is a different topic, we will save it for the

next lecture.



























