Learning rate decay

(DESCRIPTION)
Text, Optimization Algorithms. Learning rate decay. Website, deep learning, dot, A.I.

(SPEECH)
One of the things that might help speed up your learning algorithm, is to slowly reduce your learning rate over time.

We call this learning rate decay.

Let's see how you can implement

(DESCRIPTION)
New slide, Learning rate decay. Four rings of increasing size surround each other.

(SPEECH)
this.

Let's start with an example of why you might want to implement learning rate decay.

Suppose you're implementing mini-batch gradient descent, with a reasonably small mini-batch.

Maybe a mini-batch has just 64, 128 examples.

Then as you iterate, your steps will be a little bit noisy.

And it will tend towards this minimum over here, but it won't exactly converge.

But your algorithm might just end up wandering around, and never really converge, because you're using some fixed value for alpha.

And there's just some noise in your different mini-batches.

But if you were to slowly reduce your learning rate alpha, then during the initial phases, while your learning rate alpha is still large, you can still have relatively fast learning.

But then as alpha gets smaller, your steps you take will be slower and smaller.

And so you end up oscillating in a tighter region around this minimum, rather than wandering far away, even as training goes on and on.

So the intuition behind slowly reducing alpha, is that maybe during the initial steps of learning, you could afford to take much bigger steps.

But then as learning approaches converges, then having a slower learning rate allows you to take smaller steps.

So here's how you can implement learning rate decay.

Recall that one epoch is one pass, Through the data, right?

So if you have a training set as follows, maybe you break it up into different mini-batches.

Then the first pass through the training set is called the first epoch, and then the second pass is the second epoch, and so on.

So one thing you could do, is set your learning rate alpha = 1 / 1 + a parameter, which I'm going to call the decay rate, Times the epoch-num.

And this is going to be times some initial learning rate alpha 0.

Note that the decay rate here becomes another hyper-parameter, which you might need to tune.

So here's a concrete example.

If you take several epochs, so several passes through your data.

If alpha 0 = 0.2, and the decay-rate = 1, then during your first epoch, alpha will be 1 / 1 + 1 * alpha 0.

So your learning rate will be 0.1.

That's just evaluating this formula, when the decay-rate is equal to 1, and the the epoch-num is 1.

On the second epoch, your learning rate decays to 0.67.

On the third, 0.5, on the fourth, 0.4, and so on.

And feel free to evaluate more of these values yourself.

And get a sense that, as a function of your epoch number, your learning rate gradually decreases, right, according to this formula up on top.

So if you wish to use learning rate decay, what you can do, is try a variety of values of both hyper-parameter alpha 0.

As well as this decay rate hyper-parameter, and then try to find the value that works well.

Other than this formula for learning rate decay, there are a few other ways that people

(DESCRIPTION)
New slide, Other learning rate decay methods.

(SPEECH)
use.

For example, this is called exponential decay.

Where alpha is equal to some number less than 1, such as 0.95 times epoch-num, times alpha 0.

So this will exponentially quickly decay your learning rate.

Other formulas that people use are things like alpha = some constant / epoch-num square root times alpha 0.

Or some constant k, another hyper-parameter, over the mini-batch number t, square rooted, times alpha 0.

And sometimes you also see people use a learning rate that decreases in discrete steps.

Wherefore some number of steps, you have some learning rate, and then after a while you decrease it by one half.

After a while by one half.

After a while by one half.

And so this is a discrete staircase.

So so far, we've talked about using some formula to govern how alpha, the learning rate, changes over time.

One other thing that people sometimes do, is manual decay.

And so if you're training just one model at a time, and if your model takes many hours, or even many days to train.

What some people will do, is just watch your model as it's training over a large number of days.

And then manually say, it looks like the learning rate slowed down, I'm going to decrease alpha a little bit.

Of course this works, this manually controlling alpha, really tuning alpha by hand, hour by hour, or day by day.

This works only if you're training only a small number of models, but sometimes people do that as well.

So now you have a few more options for how to control the learning rate alpha.

Now, in case you're thinking, wow, this is a lot of hyper-parameters.

How do I select amongst all these different options?

I would say, don't worry about it for now.

In next week, we'll talk more about how to systematically choose hyper-parameters.

For me, I would say that learning rate decay is usually lower down on the list of things I try.

Setting alpha, just a fixed value of alpha, and getting that to be well tuned, has a huge impact.

Learning rate decay does help.

Sometimes it can really help speed up training, but it is a little bit lower down my list in terms of the things I would try.

But next week, when we talk about hyper-parameter tuning, you see more systematic ways to organize all of these hyper-parameters.

And how to efficiently search amongst them.

So that's it for learning rate decay.

Finally, I was also going to talk a little bit about local optima, and saddle points, in neural networks.

So you can have a little bit better intuition about the types of optimization problems your optimization algorithm is trying to solve, when you're trying to train these neural networks.

Let's go on to the next video to see that.