The problem of local optima

(SPEECH)
In

(DESCRIPTION)
Text, Optimization Algorithms. The problem of local optima. Website, deep learning, dot, A.I.

(SPEECH)
the early days of deep learning, people used to worry a lot about the optimization algorithm getting stuck in bad local optima.

But as this theory of deep learning has advanced, our understanding of local optima is also changing.

Let me show you how we now think about local optima and problems in the optimization problem in deep

(DESCRIPTION)
New slide, Local optima in neural networks. An illustration of a wavy surface with ridges.

(SPEECH)
learning.

This was a picture people used to have in mind when they worried about local optima.

Maybe you are trying to optimize some set of parameters, we call them W1 and W2, and the height in the surface is the cost function.

In this picture, it looks like there are a lot of local optima in all those places.

And it'd be easy for grading the sense, or one of the other algorithms to get stuck in a local optimum rather than find its way to a global optimum.

It turns out that if you are plotting a figure like this in two dimensions, then it's easy to create plots like this with a lot of different local optima.

And these very low dimensional plots used to guide their intuition.

But this intuition isn't actually correct.

It turns out if you create a neural network, most points of zero gradients are not local optima like points like this.

Instead most points of zero gradient in a cost function are saddle points.

So, that's a point where the zero gradient, again, just is maybe W1, W2, and the height is the value of the cost function J.

But informally, a function of very high dimensional space, if the gradient is zero, then in each direction it can either be a convex light function or a concave light function.

And if you are in, say, a 20,000 dimensional space, then for it to be a local optima, all 20,000 directions need to look like this.

And so the chance of that happening is maybe very small, maybe two to the minus 20,000.

Instead you're much more likely to get some directions where the curve bends up like so, as well as some directions where the curve function is bending down rather than have them all bend upwards.

So that's why in very high-dimensional spaces you're actually much more likely to run into a saddle point like that shown on the right, then the local optimum.

As for why the surface is called a saddle point, if you can picture, maybe this is a sort of saddle you put on a horse, right?

Maybe this is a horse.

This is a head of a horse, this is the eye of a horse.

Well, not a good drawing of a horse but you get the idea.

Then you, the rider, will sit here in the saddle.

That's why this point here, where the derivative is zero, that point is called a saddle point.

There's really the point on this saddle where you would sit, I guess, and that happens to have derivative zero.

And so, one of the lessons we learned in history of deep learning is that a lot of our intuitions about low-dimensional spaces, like what you can plot on the left, they really don't transfer to the very high-dimensional spaces that any other algorithms are operating over.

Because if you have 20,000 parameters, then J as your function over 20,000 dimensional vector, then you're much more likely to see saddle points than local optimum.

(DESCRIPTION)
New slide, Problem of plateaus.

(SPEECH)
If local optima aren't a problem, then what is a problem?

It turns out that plateaus can really slow down learning and a plateau is a region where the derivative is close to zero for a long time.

So if you're here, then gradient descents will move down the surface, and because the gradient is zero or near zero, the surface is quite flat.

You can actually take a very long time, you know, to slowly find your way to maybe this point on the plateau.

And then because of a random perturbation of left or right, maybe then finally I'm going to search pen colors for clarity.

Your algorithm can then find its way off the plateau.

Let it take this very long slope off before it's found its way here and they could get off this plateau.

So the takeaways from this video are, first, you're actually pretty unlikely to get stuck in bad local optima so long as you're training a reasonably large neural network, save a lot of parameters, and the cost function J is defined over a relatively high dimensional space.

But second, that plateaus are a problem and you can actually make learning pretty slow.

And this is where algorithms like momentum or RmsProp or Adam can really help your learning algorithm as well.

And these are scenarios where more sophisticated observation algorithms, such as Adam, can actually speed up the rate at which you could move down the plateau and then get off the plateau.

So because your network is solving optimizations problems over such high dimensional spaces, to be honest, I don't think anyone has great intuitions about what these spaces really look like, and our understanding of them is still evolving.

But I hope this gives you some better intuition about the challenges that the optimization algorithms may face.

So that's congratulations on coming to the end of this week's content.

Please take a look at this week's quiz as well as the [inaudible] exercise.

I hope you enjoy practicing some of these ideas of this week [inaudible] exercise and I look forward to seeing you at the start of next week's videos.