Other regularization methods

(DESCRIPTION)
Text, Regularizing your neural network. Other regularization methods. Website, deep learning, dot, A.I.

(SPEECH)
In addition to L2 regularization and drop out regularization there are few other techniques to reducing over fitting in your neural network.

Let's take a

(DESCRIPTION)
New slide, Data augmentation. A photo of an orange cat is displayedf.

(SPEECH)
look.

Let's say you fitting a CAD crossfire.

If you are over fitting getting more training data can help, but getting more training data can be expensive and sometimes you just can't get more data.

But what you can do is augment your training set by taking image like this.

And for example, flipping it horizontally and adding that also with your training set.

(DESCRIPTION)
The photo of the cat is mirrored to the side.

(SPEECH)
So now instead of just this one example in your training set, you can add this to your training example.

So by flipping the images horizontally, you could double the size of your training set.

Because you're training set is now a bit redundant this isn't as good as if you had collected an additional set of brand new independent examples.

But you could do this Without needing to pay the expense of going out to take more pictures of cats.

And then other than flipping horizontally, you can also take random crops of the image.

So here we're rotated and sort of randomly zoom into the image and this still looks like a cat.

So by taking random distortions and translations of the image you could augment your data set and make additional fake training examples.

Again, these extra fake training examples they don't add as much information as they were to call they get a brand new independent example of a cat.

But because you can do this, almost for free, other than for some confrontational costs.

This can be an inexpensive way to give your algorithm more data and therefore sort of regularize it and reduce over fitting.

And by synthesizing examples like this what you're really telling your algorithm is that If something is a cat then flipping it horizontally is still a cat.

Notice I didn't flip it vertically, because maybe we don't want upside down cats, right?

And then also maybe randomly zooming in to part of the image it's probably still a cat.

For optical character recognition you can also bring your data set by taking digits and imposing random rotations and distortions to it.

So If you add these things to your training set, these are also still digit force.

For illustration I applied a very strong distortion.

So this look very wavy for, in practice you don't need to distort the four quite as aggressively, but just a more subtle distortion than what I'm showing here, to make this example clearer for you, right?

But a more subtle distortion is usually used in practice, because this looks like really warped fours.

So data augmentation can be used as a regularization technique, in fact similar to

(DESCRIPTION)
New slide, Early stopping. A right angle graph is displayed. The horizontal plane is labeled, number of iterations.

(SPEECH)
regularization.

There's one other technique that is often used called early stopping.

So what you're going to do is as you run gradient descent you're going to plot your, either the training error, you'll use 01 classification error on the training set.

Or just plot the cost function J optimizing, and that should decrease monotonically, like so, all right?

Because as you trade, hopefully, you're trading around your cost function J should decrease.

So with early stopping, what you do is you plot this, and you also plot your dev set error.

And again, this could be a classification error in a development sense, or something like the cost function, like the logistic loss or the log loss of the dev set.

Now what you find is that your dev set error will usually go down for a while, and then it will increase from there.

So what early stopping does is, you will say well, it looks like your neural network was doing best around that iteration, so we just want to stop trading on your neural network halfway and take whatever value achieved this dev set error.

So why does this work?

Well when you've haven't run many iterations for your neural network yet your parameters w will be close to zero.

Because with random initialization you probably initialize w to small random values so before you train for a long time, w is still quite small.

And as you iterate, as you train, w will get bigger and bigger and bigger until here maybe you have a much larger value of the parameters w for your neural network.

So what early stopping does is by stopping halfway you have only a mid-size rate w.

And so similar to L2 regularization by picking a neural network with smaller norm for your parameters w, hopefully your neural network is over fitting less.

And the term early stopping refers to the fact that you're just stopping the training of your neural network earlier.

I sometimes use early stopping when training a neural network.

But it does have one downside, let me explain.

I think of the machine learning process as comprising several different steps.

One, is that you want an algorithm to optimize the cost function j and we have various tools to do that, such as grade intersect.

And then we'll talk later about other algorithms, like momentum and RMS prop and Atom and so on.

But after optimizing the cost function j, you also wanted to not over-fit.

And we have some tools to do that such as your regularization, getting more data and so on.

Now in machine learning, we already have so many hyper-parameters it surge over.

It's already very complicated to choose among the space of possible algorithms.

And so I find machine learning easier to think about when you have one set of tools for optimizing the cost function J, and when you're focusing on authorizing the cost function J.

All you care about is finding w and b, so that J(w,b) is as small as possible.

You just don't think about anything else other than reducing this.

And then it's completely separate task to not over fit, in other words, to reduce variance.

And when you're doing that, you have a separate set of tools for doing it.

And this principle is sometimes called orthogonalization.

And there's this idea, that you want to be able to think about one task at a time.

I'll say more about orthorganization in a later video, so if you don't fully get the concept yet, don't worry about it.

But, to me the main downside of early stopping is that this couples these two tasks.

So you no longer can work on these two problems independently, because by stopping gradient decent early, you're sort of breaking whatever you're doing to optimize cost function J, because now you're not doing a great job reducing the cost function J.

You've sort of not done that that well.

And then you also simultaneously trying to not over fit.

So instead of using different tools to solve the two problems, you're using one that kind of mixes the two.

And this just makes the set of things you could try are more complicated to think about.

Rather than using early stopping, one alternative is just use L2 regularization then you can just train the neural network as long as possible.

I find that this makes the search space of hyper parameters easier to decompose, and easier to search over.

But the downside of this though is that you might have to try a lot of values of the regularization parameter lambda.

And so this makes searching over many values of lambda more computationally expensive.

And the advantage of early stopping is that running the gradient descent process just once, you get to try out values of small w, mid-size w, and large w, without needing to try a lot of values of the L2 regularization hyperparameter lambda.

If this concept doesn't completely make sense to you yet, don't worry about it.

We're going to talk about orthogonalization in greater detail in a later video, I think this will make a bit more sense.

Despite it's disadvantages, many people do use it.

I personally prefer to just use L2 regularization and try different values of lambda.

That's assuming you can afford the computation to do so.

But early stopping does let you get a similar effect without needing to explicitly try lots of different values of lambda.

So you've now seen how to use data augmentation as well as if you wish early stopping in order to reduce variance or prevent over fitting your neural network.

Next let's talk about some techniques for setting up your optimization problem to make your training go quickly.