Why regularization reduces overfitting?

(SPEECH)
Why

(DESCRIPTION)
Text, Regularizing your neural network. Why regularization reduces overfitting. Website, deep learning, dot, A.I.

(SPEECH)
does regularization help with overfitting?

Why does it help with reducing variance problems?

Let's go through a couple examples to gain some intuition about

(DESCRIPTION)
New slide, How does regularization prevent overfitting?

(SPEECH)
how it works.

So, recall that high bias, high variance.

And I just write pictures from our earlier video that looks something like this.

Now, let's see a fitting large and deep neural network.

I know I haven't drawn this one too large or too deep, unless you think some neural network and this currently overfitting.

So you have some cost function like J of W, B equals sum of the losses.

So what we did for regularization was add this extra term that penalizes the weight matrices from being too large.

So that was the Frobenius norm.

So why is it that shrinking the L two norm or the Frobenius norm or the parameters might cause less overfitting?

One piece of intuition is that if you crank regularisation lambda to be really, really big, they'll be really incentivized to set the weight matrices W to be reasonably close to zero.

So one piece of intuition is maybe it set the weight to be so close to zero for a lot of hidden units that's basically zeroing out a lot of the impact of these hidden units.

And if that's the case, then this much simplified neural network becomes a much smaller neural network.

In fact, it is almost like a logistic regression unit, but stacked most probably as deep.

And so that will take you from this overfitting case much closer to the left to other high bias case.

But hopefully there'll be an intermediate value of lambda that results in a result closer to this just right case in the middle.

But the intuition is that by cranking up lambda to be really big they'll set W close to zero, which in practice this isn't actually what happens.

We can think of it as zeroing out or at least reducing the impact of a lot of the hidden units so you end up with what might feel like a simpler network.

They get closer and closer as if you're just using logistic regression.

The intuition of completely zeroing out of a bunch of hidden units isn't quite right.

It turns out that what actually happens is they'll still use all the hidden units, but each of them would just have a much smaller effect.

But you do end up with a simpler network and as if you have a smaller network that is therefore less prone to overfitting.

So a lot of this intuition helps better when you implement regularization in the program exercise, you actually see some of these variance reduction results yourself.

Here's another attempt at additional intuition for why regularization helps prevent overfitting.

And for this, I'm going to assume that we're using the tanh activation function which looks like this.

This is a g of z equals tanh of z.

So if that's the case, notice that so long as Z is quite small, so if Z takes on only a smallish range of parameters, maybe around here, then you're just using the linear regime of the tanh function.

Is only if Z is allowed to wander up to larger values or smaller values like so, that the activation function starts to become less linear.

So the intuition you might take away from this is that if lambda, the regularization parameter, is large, then you have that your parameters will be relatively small, because they are penalized being large into a cos function.

And so if the blades W are small then because Z is equal to W and then technically is plus b, but if W tends to be very small, then Z will also be relatively small.

And in particular, if Z ends up taking relatively small values, just in this whole range, then G of Z will be roughly linear.

So it's as if every layer will be roughly linear.

As if it is just linear regression.

And we saw in course one that if every layer is linear then your whole network is just a linear network.

And so even a very deep network, with a deep network with a linear activation function is at the end they are only able to compute a linear function.

So it's not able to fit those very very complicated decision.

Very non-linear decision boundaries that allow it to really overfit right to data sets like we saw on the overfitting high variance case on the previous slide.

So just to summarize, if the regularization becomes very large, the parameters W very small, so Z will be relatively small, kind of ignoring the effects of b for now, so Z will be relatively small or, really, I should say it takes on a small range of values.

And so the activation function if is tanh, say, will be relatively linear.

And so your whole neural network will be computing something not too far from a big linear function which is therefore pretty simple function rather than a very complex highly non-linear function.

And so is also much less able to overfit.

And again, when you enter in regularization for yourself in the program exercise, you'll be able to see some of these effects yourself.

Before wrapping up our def discussion on regularization, I just want to give you one implementational tip.

Which is that, when implanting regularization, we took our definition of the cost function J and we actually modified it by adding this extra term that penalizes the weight being too large.

And so if you implement gradient descent, one of the steps to debug gradient descent is to plot the cost function J as a function of the number of elevations of gradient descent and you want to see that the cost function J decreases monotonically after every elevation of gradient descent.

And if you're implementing regularization then please remember that J now has this new definition.

If you plot the old definition of J, just this first term, then you might not see a decrease monotonically.

So to debug gradient descent make sure that you're plotting this new definition of J that includes this second term as well.

Otherwise you might not see J decrease monotonically on every single elevation.

So that's it for L two regularization which is actually a regularization technique that I use the most in training deep learning modules.

In deep learning there is another sometimes used regularization technique called dropout regularization.

Let's take a look at that in the next video.