Hyperparameters tuning in practice Pandas vs Caviar

(DESCRIPTION)
Text, Hyperparameters tuning. Hyperparameters tuning in practice: Pandas vs. Caviar. Website, deep learning, dot, A.I.

(SPEECH)
You have now heard a lot about how to search for good hyperparameters.

Before wrapping up our discussion on hyperparameter search, I want to share with you just a couple of final tips and tricks for how to organize your hyperparameter search process.

(DESCRIPTION)
New slide, Re-test hyperparameters occasionally. A self-containing circle is illustrated with three functions, idea, code, and experiment.

(SPEECH)
Deep learning today is applied to many different application areas and that intuitions about hyperparameter settings from one application area may or may not transfer to a different one.

There is a lot of cross-fertilization among different applications' domains, so for example, I've seen ideas developed in the computer vision community, such as Confonets or ResNets, which we'll talk about in a later course, successfully applied to speech.

I've seen ideas that were first developed in speech successfully applied in NLP, and so on.

So one nice development in deep learning is that people from different application domains do read increasingly research papers from other application domains to look for inspiration for cross-fertilization.

In terms of your settings for the hyperparameters, though, I've seen that intuitions do get stale.

So even if you work on just one problem, say logistics, you might have found a good setting for the hyperparameters and kept on developing your algorithm, or maybe seen your data gradually change over the course of several months, or maybe just upgraded servers in your data center.

And because of those changes, the best setting of your hyperparameters can get stale.

So I recommend maybe just retesting or reevaluating your hyperparameters at least once every several months to make sure that you're still happy with the values you

(DESCRIPTION)
New slide, Babysitting one model.

(SPEECH)
have.

Finally, in terms of how people go about searching for hyperparameters, I see maybe two major schools of thought, or maybe two major different ways in which people go about it.

One way is if you babysit one model.

And usually you do this if you have maybe a huge data set but not a lot of computational resources, not a lot of CPUs and GPUs, so you can basically afford to train only one model or a very small number of models at a time.

In that case you might gradually babysit that model even as it's training.

So, for example, on Day 0 you might initialize your parameter as random and then start training.

And you gradually watch your learning curve, maybe the cost function J or your dataset error or something else, gradually decrease over the first day.

Then at the end of day one, you might say, gee, looks it's learning quite well, I'm going to try increasing the learning rate a little bit and see how it does.

And then maybe it does better.

And then that's your Day 2 performance.

And after two days you say, okay, it's still doing quite well.

Maybe I'll fill the momentum term a bit or decrease the learning variable a bit now, and then you're now into Day 3.

And every day you kind of look at it and try nudging up and down your parameters.

And maybe on one day you found your learning rate was too big.

So you might go back to the previous day's model, and so on.

But you're kind of babysitting the model one day at a time even as it's training over a course of many days or over the course of several different weeks.

So that's one approach, and people that babysit one model, that is watching performance and patiently nudging the learning rate up or down.

But that's usually what happens if you don't have enough computational capacity to train a lot of models at the same

(DESCRIPTION)
A second graph is displayed, titled Training many models in parallel.

(SPEECH)
time.

The other approach would be if you train many models in parallel.

So you might have some setting of the hyperparameters and just let it run by itself ,either for a day or even for multiple days, and then you get some learning curve like that; and this could be a plot of the cost function J or cost of your training error or cost of your dataset error, but some metric in your tracking.

And then at the same time you might start up a different model with a different setting of the hyperparameters.

And so, your second model might generate a different learning curve, maybe one that looks like that.

I will say that one looks better.

And at the same time, you might train a third model, which might generate a learning curve that looks like that, and another one that, maybe this one diverges so it looks like that, and so on.

Or you might train many different models in parallel, where these orange lines are different models, right, and so this way you can try a lot of different hyperparameter settings and then just maybe quickly at the end pick the one that works best.

Looks like in this example it was, maybe this curve that look best.

So to make an analogy, I'm going to call the approach on the left the panda approach.

(DESCRIPTION)
A photo of a panda bear appears in the bottom left corner.

(SPEECH)
When pandas have children, they have very few children, usually one child at a time, and then they really put a lot of effort into making sure that the baby panda survives.

So that's really babysitting.

One model or one baby panda.

(DESCRIPTION)
New slide, A photo of caviar appears in the bottom right corner.

(SPEECH)
Whereas the approach on the right is more like what fish do.

I'm going to call this the caviar strategy.

There's some fish that lay over 100 million eggs in one mating season.

But the way fish reproduce is they lay a lot of eggs and don't pay too much attention to any one of them but just see that hopefully one of them, or maybe a bunch of them, will do well.

So I guess, this is really the difference between how mammals reproduce versus how fish and a lot of reptiles reproduce.

But I'm going to call it the panda approach versus the caviar approach, since that's more fun and memorable.

So the way to choose between these two approaches is really a function of how much computational resources you have.

If you have enough computers to train a lot of models in parallel, then by all means take the caviar approach and try a lot of different hyperparameters and see what works.

But in some application domains, I see this in some online advertising settings as well as in some computer vision applications, where there's just so much data and the models you want to train are so big that it's difficult to train a lot of models at the same time.

It's really application dependent of course, but I've seen those communities use the panda approach a little bit more, where you are kind of babying a single model along and nudging the parameters up and down and trying to make this one model work.

Although, of course, even the panda approach, having trained one model and then seen it work or not work, maybe in the second week or the third week, maybe I should initialize a different model and then baby that one along just like even pandas, I guess, can have multiple children in their lifetime, even if they have only one, or a very small number of children, at any one time.

So hopefully this gives you a good sense of how to go about the hyperparameter search process.

Now, it turns out that there's one other technique that can make your neural network much more robust to the choice of hyperparameters.

It doesn't work for all neural networks, but when it does, it can make the hyperparameter search much easier and also make training go much faster.

Let's talk about this technique in the next video.