Orthogonalization

(DESCRIPTION)
Text, introduction to ML strategy. Orthogonalization

(SPEECH)
One of the challenges with building machine learning systems is that there's so many things you could try, so many things you could change.

Including, for example, so many hyperparameters you could tune.

One of the things I've noticed is about the most effective machine learning people is they're very clear-eyed about what to tune in order to try to achieve one effect.

This is a process we call orthogonalization.

Let me tell you what I mean.

(DESCRIPTION)
Text, TV tuning example. Image of TV from the 1950s

(SPEECH)
Here's a picture of an old school television, with a lot of knobs that you could tune to adjust the picture in various ways.

So for these old TV sets, maybe there was one knob to adjust how tall vertically your image is and another knob to adjust how wide it is.

Maybe another knob to adjust how trapezoidal it is, another knob to adjust how much to move the picture left and right, another one to adjust how much the picture's rotated, and so on.

(DESCRIPTION)
Diagrams illustrating effect of each knob

(SPEECH)
And what TV designers had spent a lot of time doing was to build the circuitry, really often analog circuitry back then, to make sure each of the knobs had a relatively interpretable function.

Such as one knob to tune this, one knob to tune this, one knob to tune this, and so on.

In contrast, imagine if you had a knob that tunes 0.1 x how tall the image is, + 0.3 x how wide the image is,- 1.7 x how trapezoidal the image is, + 0.8 times the position of the image on the horizontal axis, and so on.

If you tune this knob, then the height of the image, the width of the image, how trapezoidal it is, how much it shifts, it all changes all at the same time.

If you have a knob like that, it'd be almost impossible to tune the TV so that the picture gets centered in the display area.

So in this context, orthogonalization refers to that the TV designers had designed the knobs so that each knob kind of does only one thing.

And this makes it much easier to tune the TV, so that the picture gets centered where you want it to be.

(DESCRIPTION)
Drawing a centered rectangle on the TV screen

(SPEECH)
Here's another example of orthogonalization.

(DESCRIPTION)
Car driver's seat and dashboard

(SPEECH)
If you think about learning to drive a car, a car has three main controls, which are steering, the steering wheel decides how much you go left or right, acceleration, and braking.

So these three controls, or really one control for steering and another two controls for your speed.

It makes it relatively interpretable, what your different actions through different controls will do to your car.

But now imagine if someone were to build a car so that there was a joystick, where one axis of the joystick controls 0.3 x your steering angle,- 0.8 x your speed.

And you had a different control that controls 2 x the steering angle, + 0.9 x the speed of your car.

In theory, by tuning these two knobs, you could get your car to steer at the angle and at the speed you want.

But it's much harder than if you had just one single control for controlling the steering angle, and a separate, distinct set of controls for controlling the speed.

(DESCRIPTION)
Two-dimensional graph. One axis represents angle, the other perpendicular axis represents speed

(SPEECH)
So the concept of orthogonalization refers to that, if you think of one dimension of what you want to do as controlling a steering angle, and another dimension as controlling your speed.

Then you want one knob to just affect the steering angle as much as possible, and another knob, in the case of the car, is really acceleration and braking, that controls your speed.

But if you had a control that mixes the two together, like a control like this one that affects both your steering angle and your speed, something that changes both at the same time, then it becomes much harder to set the car to the speed and angle you want.

And by having orthogonal, orthogonal means at 90 degrees to each other.

By having orthogonal controls that are ideally aligned with the things you actually want to control, it makes it much easier to tune the knobs you have to tune.

To tune the steering wheel angle, and your accelerator, your braking, to get the car to do what you want.

So how does this relate to machine learning?

(DESCRIPTION)
Text, chain of assumptions in ML. Fit training set well on cost function

(SPEECH)
For a supervised learning system to do well, you usually need to tune the knobs of your system to make sure that four things hold true.

First, is that you usually have to make sure that you're at least doing well on the training set.

So performance on the training set needs to pass some acceptability assessment.

For some applications, this might mean doing comparably to human level performance.

But this will depend on your application, and we'll talk more about comparing to human level performance next week.

(DESCRIPTION)
Text, fit dev set well on cost function

(SPEECH)
But after doing well on the training sets, you then hope that this leads to also doing well on the dev set.

And

(DESCRIPTION)
Text, fit test set well on cost function

(SPEECH)
you then hope that this also does well on the test set.

And

(DESCRIPTION)
Text, performs well in real world

(SPEECH)
finally, you hope that doing well on the test set on the cost function results in your system performing in the real world.

So you hope that this resolves in happy cat picture app users, for example.

So to relate back to the TV tuning example, if the picture of your TV was either too wide or too narrow, you wanted one knob to tune in order to adjust that.

You don't want to have to carefully adjust five different knobs, which also affect different things.

You want one knob to just affect the width of your TV image.

So

(DESCRIPTION)
Drawing a cylindrical knob in rotation

(SPEECH)
in a similar way, if your algorithm is not fitting the training set well on the cost function, you want one knob, yes, that's my attempt to draw a knob.

Or maybe one specific set of knobs that you can use, to make sure you can tune your algorithm to make it fit well on the training set.

So the knobs you use to tune this are, you might train a bigger network.

Or you might switch to a better optimization algorithm, like the Adam optimization algorithm, and so on, into some other options we'll discuss later this week and next week.

In contrast, if you find that the algorithm is not fitting the dev set well, then there's a separate set of knobs.

Yes, that's my not very artistic rendering of another knob, you want to have a distinct set of knobs to try.

So for example, if your algorithm is not doing well on the dev set, it's doing well on the training set but not on the dev set, then you have a set of knobs around regularization that you can use to try to make it satisfy the second criteria.

So the analogy is, now that you've tuned the width of your TV set, if the height of the image isn't quite right, then you want a different knob in order to tune the height of the TV image.

And you want to do this hopefully without affecting the width of your TV image too much.

And getting a bigger training set would be another knob you could use, that helps your learning algorithm generalize better to the dev set.

Now, having adjusted the width and height of your TV image, well, what if it doesn't meet the third criteria?

What if you do well on the dev set but not on the test set?

If that happens, then the knob you tune is, you probably want to get a bigger dev set.

Because if it does well on the dev set but not the test set, it probably means you've overtuned to your dev set, and you need to go back and find a bigger dev set.

And finally, if it does well on the test set, but it isn't delivering to you a happy cat picture app user, then what that means is that you want to go back and change either the dev set or the cost function.

Because if doing well on the test set according to some cost function doesn't correspond to your algorithm doing what you need it to do in the real world, then it means that either your dev test set distribution isn't set correctly, or your cost function isn't measuring the right thing.

I know I'm going over these examples quite quickly, but we'll go much more into detail on these specific knobs later this week and next week.

So if you aren't following all the details right now, don't worry about it.

But I want to give you a sense of this orthogonalization process, that you want to be very clear about which of these maybe four issues, the different things you could tune, are trying to address.

And

(DESCRIPTION)
In red ink, text, early stopping. Arrows pointing from early stopping to two different elements in the chain: fit of the training set, and fit of the dev set

(SPEECH)
when I train a neural network, I tend not to use early stopping.

It's not a bad technique, quite a lot of people do it.

But I personally find early stopping difficult to think about.

Because this is an op that simultaneously affects how well you fit the training set, because if you stop early, you fit the training set less well.

It also simultaneously is often done to improve your dev set performance.

So this is one knob that is less orthogonalized, because it simultaneously affects two things.

It's like a knob that simultaneously affects both the width and the height of your TV image.

And it doesn't mean that it's bad, not to use, you can use it if you want.

But when you have more orthogonalized controls, such as these other ones that I'm writing down here, then it just makes the process of tuning your network much easier.

So I hope that gives you a sense of what orthogonalization means.

Just like when you look at the TV image, it's nice if you can say, my TV image is too wide, so I'm going to tune this knob, or it's too tall, so I'm going to tune that knob, or it's too trapezoidal, so I'm going to have to tune that knob.

In machine learning, it's nice if you can look at your system and say, this piece of it is wrong.

It does not do well on the training set, it does not do well on the dev set, it does not do well on the test set, or it's doing well on the test set but just not in the real world.

But figure out exactly what's wrong, and then have exactly one knob, or a specific set of knobs that helps to just solve that problem that is limiting the performance of machine learning system.

So what we're going to do this week and next week is go through how to diagnose what exactly is the bottleneck to your system's performance.

As well as identify the specific set of knobs you could use to tune your system to improve that aspect of its performance.

So let's start going more into the details of this process.