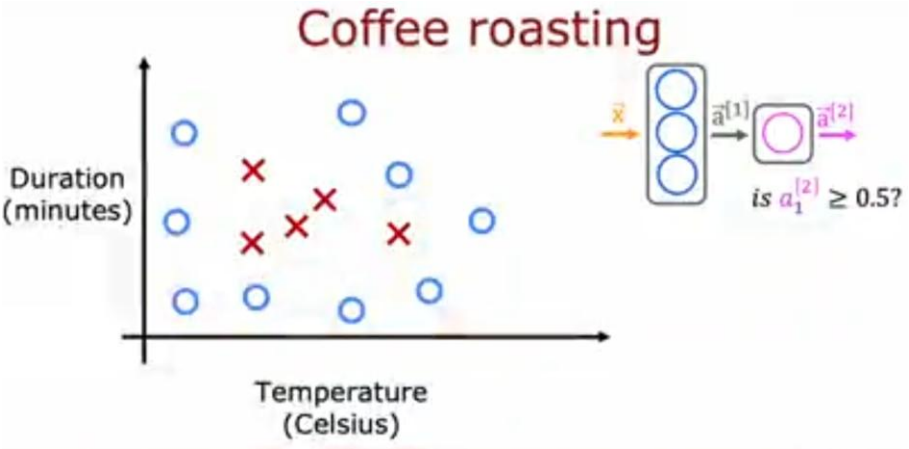
Inference in code

TensorFlow is one of the leading frameworks to implement deep learning algorithms. When I'm building projects, TensorFlow is actually a tool that I use most often. The other popular tool is PyTorch. But we're going to focus in this specialization on TensorFlow.

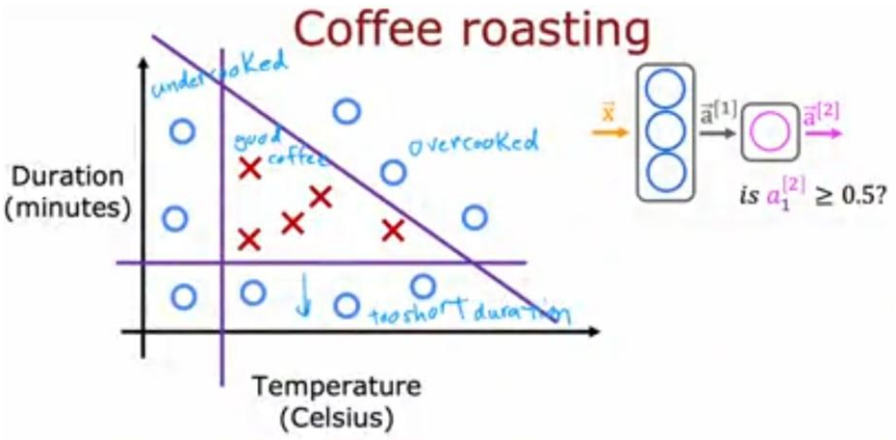
In this video, let's take a look at how you can implement inferencing code using TensorFlow. Let's dive in. One of the remarkable things about neural networks is the same algorithm can be applied to so many different applications.



For this video and in some of the labs for you to see what the neural network is doing, I'm going to use another example to illustrate inference. Sometimes I do like to roast coffee beans myself at home. My favorite is actually Colombian coffee beans. Can the learning algorithm help optimize the quality of the beans you get from a roasting process like this?

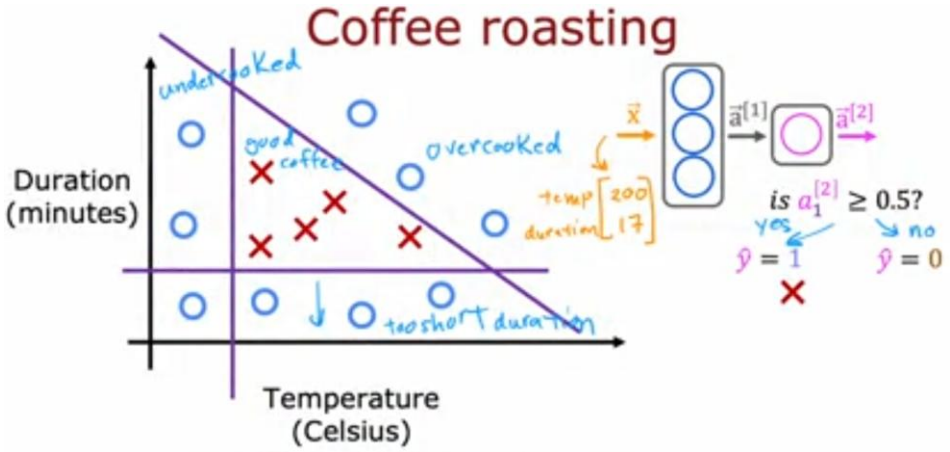
When you're roasting coffee, two parameters you get to control are the temperature at which you're heating up the raw coffee beans to turn them into nicely roasted coffee beans, as well as the duration or how long are you going to roast the beans.

In this slightly simplified example, we've created the datasets of different temperatures and different durations, as well as labels showing whether the coffee you roasted is good-tasting coffee. Where cross here, the positive cross y equals 1 corresponds to good coffee, and all the negative cross corresponds to bad coffee.



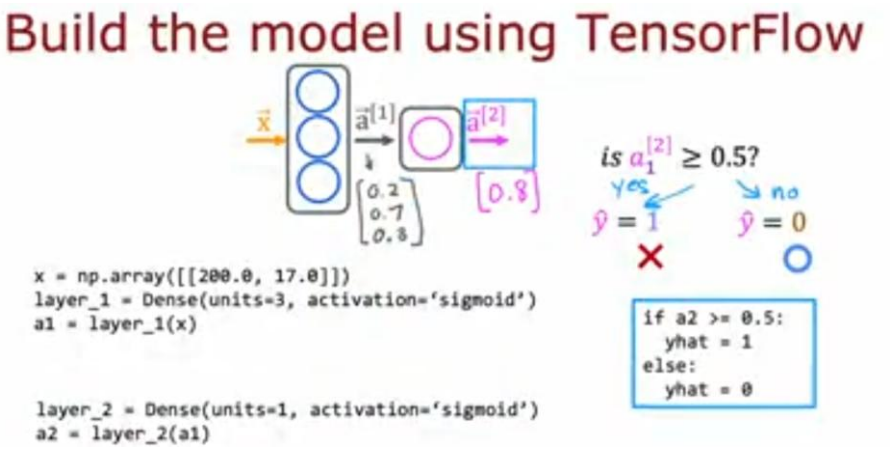
It looks like a reasonable way to think of this dataset is if you cook it at too lower temperature, it doesn't get roasted and it ends up undercooked. If you cook it, not for long enough, the duration is too short, it's also not a nicely roasted set of beans.

Finally, if you were to cook it either for too long or for too higher temperature, then you end up with overcooked beans. They're a little bit burnt beans. There's not good coffee either. It's only points within this little triangle here that corresponds to good coffee. This example is simplified a bit from actual coffee roasting.



Even though this example is a simplified one for the purpose of illustration, there have actually been serious projects using machine learning to optimize coffee roasting as well. The task is given a feature vector x with both temperature and duration, say 200 degrees Celsius for 17 minutes, how can we do inference in a neural network to get it to tell us whether or not this temperature and duration setting will result in good coffee or not?

It looks like this. We're going to set x to be an array of two numbers. The input features 200 degrees celsius and 17 minutes. Then you create Layer 1 as this first hidden layer, the neural network, as dense open parentheses units 3, that means three units or three hidden units in this layer using as the activation function, the sigmoid function. Dense is another name for the layers of a neural network that we've learned about so far.

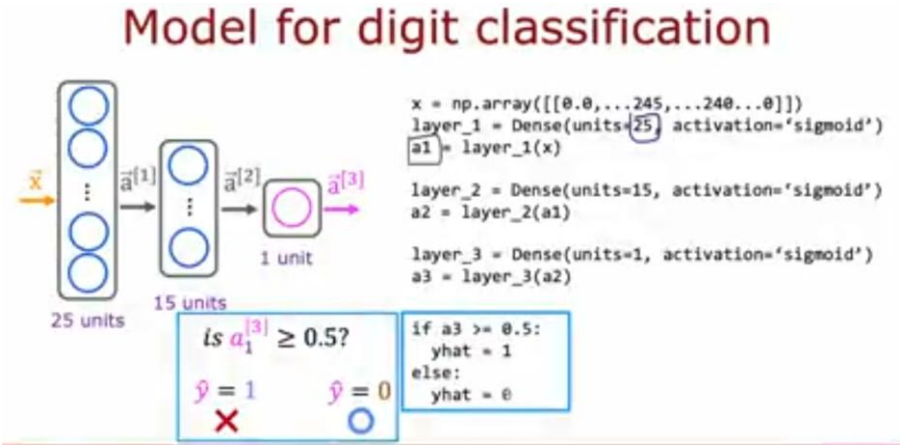


As you learn more about neural networks, you learn about other types of layers as well. But for now, we'll just use the dense layer, which is the layer type you've learned about in the last few videos for all of our examples. Next, you compute a1 by taking Layer 1, which is actually a function, and applying this function Layer 1 to the values of x.

That's how you get a1, which is going to be a list of three numbers because Layer 1 had three units. So a1 here may, just for the sake of illustration, be 0.2, 0.7, 0.3. Next, for the second hidden layer, Layer 2, would be dense. Now this time it has one unit and again to sigmoid activation function, and you can then compute a2 by applying this Layer 2 function to the activation values from Layer 1 to a1.

That will give you the value of a2, which for the sake of illustration is maybe 0.8. Finally, if you wish to threshold it at 0.5, then you can just test if a2 is greater and equal to 0.5 and set y-hat equals to one or zero positive or negative cross accordingly. That's how you do inference in the neural network using TensorFlow.

There are some additional details that I didn't go over here, such as how to load the TensorFlow library and how to also load the parameters w and b of the neural network. But we'll go over that in the lab. Please be sure to take a look at the lab. But these are the key steps for forward propagation in how you compute a1 and a2 and optionally threshold a2.



Let's look at one more example and we're going to go back to the handwritten digit classification problem. In this example, x is a list of the pixel intensity values. So x is equal to a numpy array of this list of pixel intensity values. Then to initialize and carry out one step of forward propagation, Layer 1 is a dense layer with 25 units and the sigmoid activation function.

You then compute a1 equals the Layer 1 function applied to x. To build and carry out inference through the second layer, similarly, you set up Layer 2 as follows, and then computes a2 as Layer 2 applied to a1. Then finally, Layer 3 is the third and final dense layer.

Then finally, you can optionally threshold a3 to come up with a binary prediction for y-hat. That's the syntax for carrying out inference in TensorFlow. One thing I briefly alluded to is the structure of the numpy arrays. TensorFlow treats data in a certain way that is important to get right.