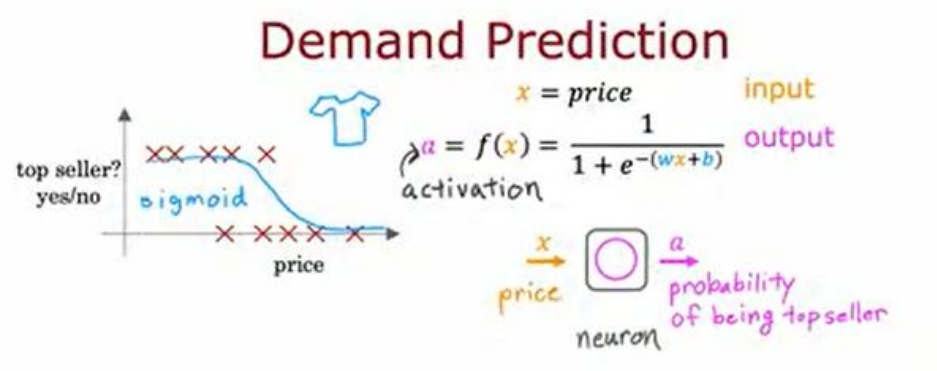
Prévision de la demande

To illustrate how neural networks work, let's start with an example. We'll use an example from demand prediction in which you look at the product and try to predict, will this product be a top seller or not? Let's take a look. In this example, you're selling T-shirts and you would like to know if a particular T-shirt will be a top seller, yes or no, and you have collected data of different t-shirts that were sold at different prices, as well as which ones became a top seller.

This type of application is used by retailers today in order to plan better inventory levels as well as marketing campaigns. If you know what's likely to be a top seller, you would plan, for example, to just purchase more of that stock in advance. In this example, the input feature x is the price of the T-shirt, and so that's the input to the learning algorithm.



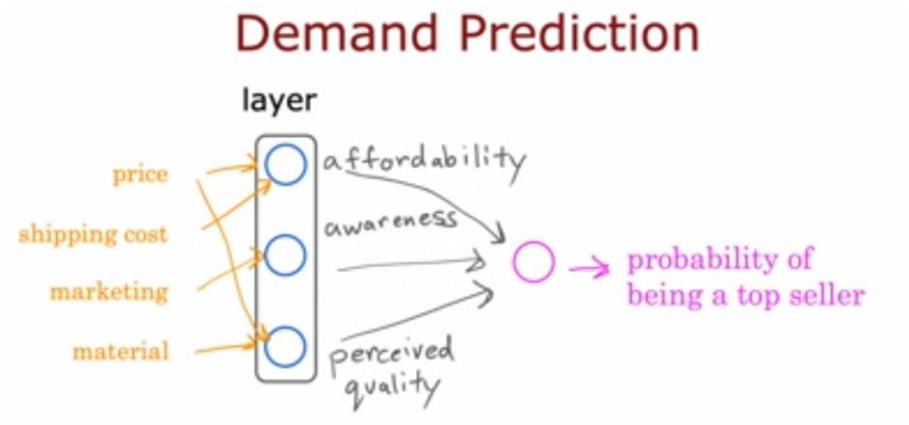
If you apply logistic regression to fit a sigmoid function to the data that might look like that then the outputs of your prediction might look like this, 1/1 plus e to the negative wx plus b. Previously, we had written this as f of x as the output of the learning algorithm. In order to set us up to build a neural network, I'm going to switch the terminology a little bit and use the alphabet a to denote the output of this logistic regression algorithm.

The term a stands for activation, and it's actually a term from neuroscience, and it refers to how much a neuron is sending a high output to other neurons downstream from it. It turns out that this logistic regression units or this little logistic regression algorithm, can be thought of as a very simplified model of a single neuron in the brain.

Where what the neuron does is it takes us input the price x, and then it computes this formula on top, and it outputs the number a, which is computed by this formula, and it outputs the probability of this T-shirt being a top seller. Another way to think of a neuron is as a tiny little computer whose only job is to input one number or a few numbers, such as a price, and then to output one number or maybe a few other numbers which in this case is the probability of the T-shirt being a top seller.

As I alluded in the previous video, a logistic regression algorithm is much simpler than what any biological neuron in your brain or mine does. Which is why the artificial neural network is such a vastly oversimplified model of the human brain. Even though in practice, as you know, deep learning algorithms do work very well.

Given this description of a single neuron, building a neural network now it just requires taking a bunch of these neurons and wiring them together or putting them together. Let's now look at a more complex example of demand prediction. In this example, we're going to have four features to predict whether or not a T-shirt is a top seller. The features are the price of the T-shirt, the shipping costs, the amounts of marketing of that particular T-shirt, as well as the material quality, is this a high-quality, thick cotton versus maybe a lower quality material?



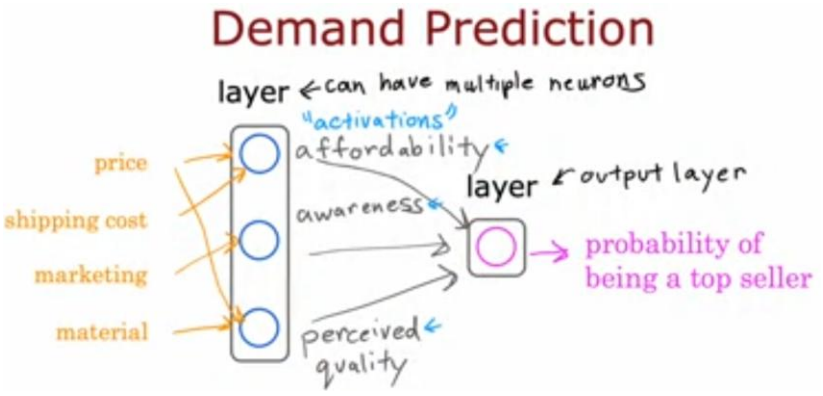
Now, you might suspect that whether or not a T-shirt becomes a top seller actually depends on a few factors. First, one is the affordability of this T-shirt. Second is, what's the degree of awareness of this T-shirt that potential buyers have? Third is perceived quality to bias or potential bias saying this is a high-quality T-shirt. What I'm going to do is create one artificial neuron to try to estimate the probability that this T-shirt is perceive as highly affordable.

Affordability is mainly a function of price and shipping costs because the total amount of the pay is some of the price plus the shipping costs. We're going to use a little neuron here, a logistic regression unit to input price and shipping costs and predict do people think this is affordable?

Second, I'm going to create another artificial neuron here to estimate, is there high awareness of this? Awareness in this case is mainly a function of the marketing of the T-shirt. Finally, going to create another neuron to estimate do people perceive this to be of high quality, and that may mainly be a function of the price of the T-shirt and of the material quality.

Price is a factor here because fortunately or unfortunately, if there's a very high priced T-shirt, people will sometimes perceive that to be of high quality because it is very expensive than maybe people think it's going to be of high-quality. Given these estimates of affordability, awareness, and perceived quality we then wire the outputs of these three neurons to another neuron here on the right, that then there's another logistic regression unit.

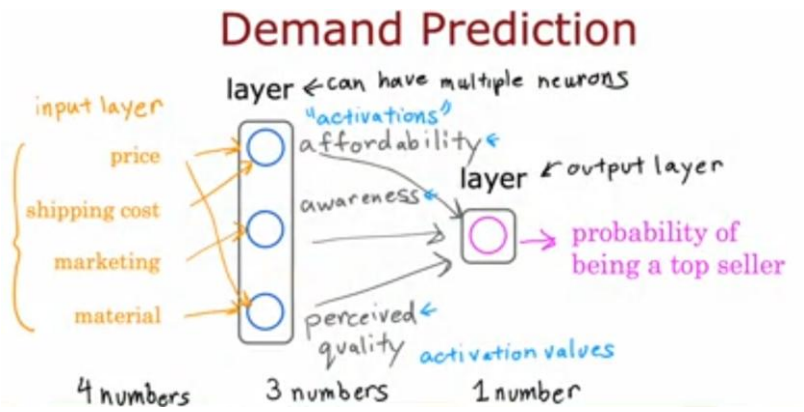
That finally inputs those three numbers and outputs the probability of this t-shirt being a top seller. In the terminology of neural networks, we're going to group these three neurons together into what's called a layer.



A layer is a grouping of neurons which takes as input the same or similar features, and that in turn outputs a few numbers together. These three neurons on the left form one layer which is why I drew them on top of each other, and this single neuron on the right is also one layer. The layer on the left has three neurons, so a layer can have multiple neurons or it can also have a single neuron as in the case of this layer on the right.

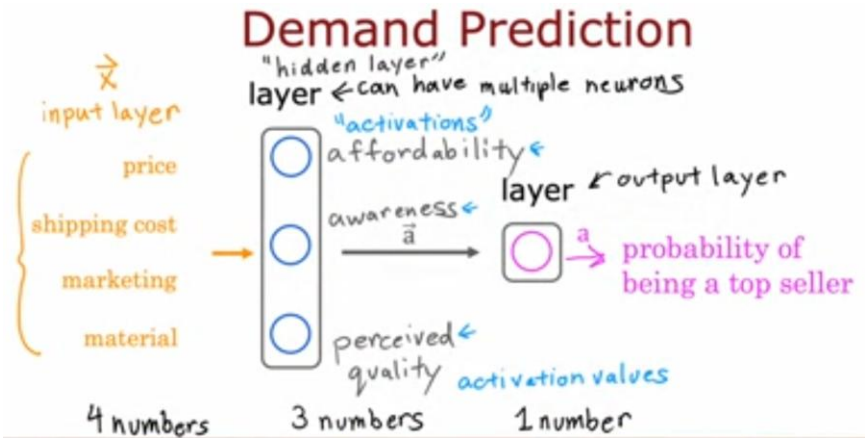
This layer on the right is also called the output layer because the outputs of this final neuron is the output probability predicted by the neural network. In the terminology of neural networks we're also going to call affordability awareness and perceive quality to be activations. The term activations comes from biological neurons, and it refers to the degree that the biological neuron is sending a high output value or sending many electrical impulses to other neurons to the downstream from it.

These numbers on affordability, awareness, and perceived quality are the activations of these three neurons in this layer, and also this output probability is the activation of this neuron shown here on the right.



This particular neural network therefore carries out computations as follows. It inputs four numbers then this layer of the neural network uses those four numbers to compute the new numbers also called activation values. Then the final layer, the output layer of the neural network used those three numbers to compute one number. In a neural network this list of four numbers is also called the input layer, and that's just a list of four numbers.

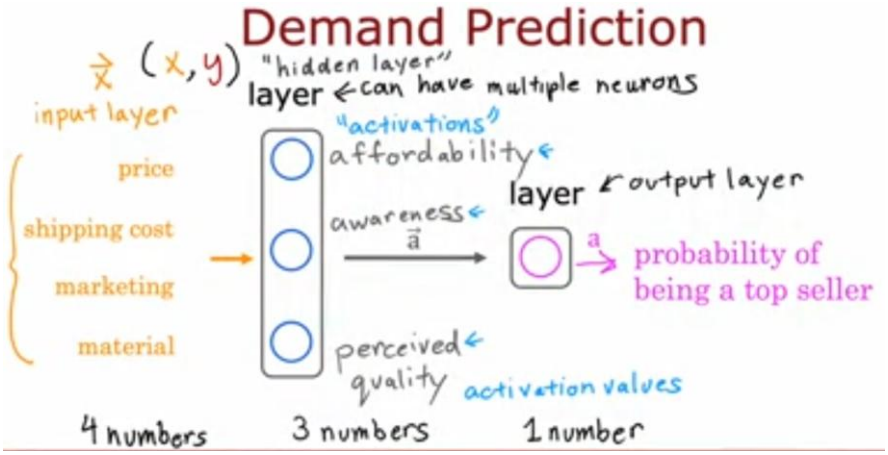
Now, there's one simplification I'd like make to this neural network. The way I've described it so far, we had to go through the neurons one at a time and decide what inputs it would take from the previous layer. For example, we said affordability is a function of just price and shipping costs and awareness is a function of just marketing and so on, but if you're building a large neural network it'd be a lot of work to go through and manually decide which neurons should take which features as inputs.



The way a neural network is implemented in practice each neuron in a certain layer; say this layer in the middle, will have access to every feature, to every value from the previous layer, from the input layer which is why I'm now drawing arrows from every input feature to every one of these neurons shown here in the middle.

You can imagine that if you're trying to predict affordability and it knows what's the price shipping cost marketing and material, may be you'll learn to ignore marketing and material and just figure out through setting the parameters appropriately to only focus on the subset of features that are most relevant to affordability. To further simplify the notation and the description of this neural network I'm going to take these four input features and write them as a vector x, and we're going to view the neural network as having four features that comprise this feature vector x.

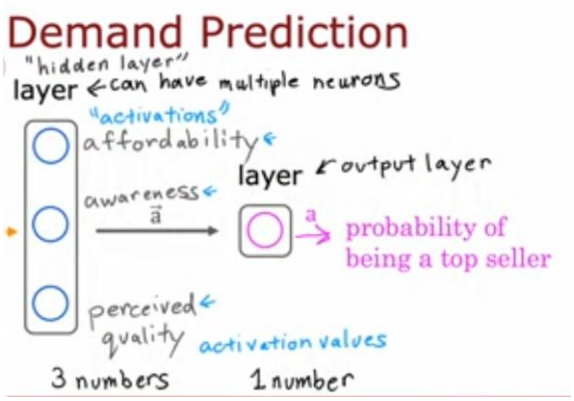
This feature vector is fed to this layer in the middle which then computes three activation values. That is these numbers and these three activation values in turn becomes another vector which is fed to this final output layer that finally outputs the probability of this t-shirt to being a top seller. That's all a neural network is. It has a few layers where each layer inputs a vector and outputs another vector of numbers.



For example, this layer in the middle inputs four numbers x and outputs three numbers corresponding to affordability, awareness, and perceived quality. To add a little bit more terminology, you've seen that this layer is called the output layer and this layer is called the input layer.

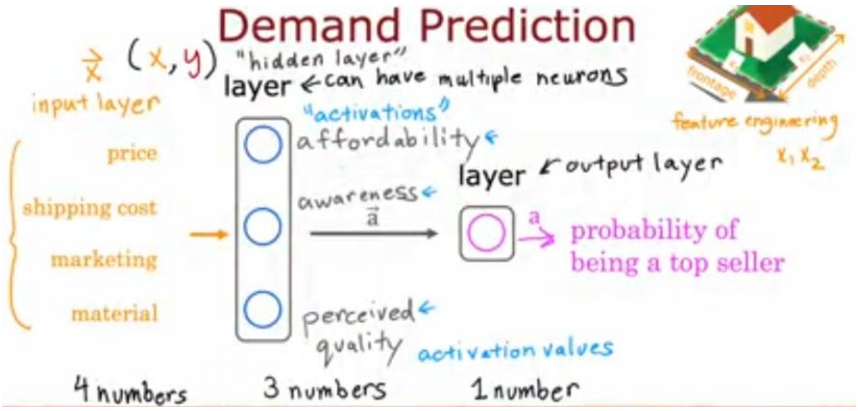
To give the layer in the middle a name as well, this layer in the middle is called a hidden layer. I know that this is maybe not the best or the most intuitive name but that terminology comes from that's when you have a training set. In a training set, you get to observe both x and y. Your data set tells you

what is x and what is y, and so you get data that tells you what are the correct inputs and the correct outputs. But your dataset doesn't tell you what are the correct values for affordability, awareness, and perceived quality.



The correct values for those are hidden. You don't see them in the training set, which is why this layer in the middle is called a hidden layer. I'd like to share with you another way of thinking about neural networks that I've found useful for building my intuition about it. Just let me cover up the left half of this diagram, and see what we're left with.

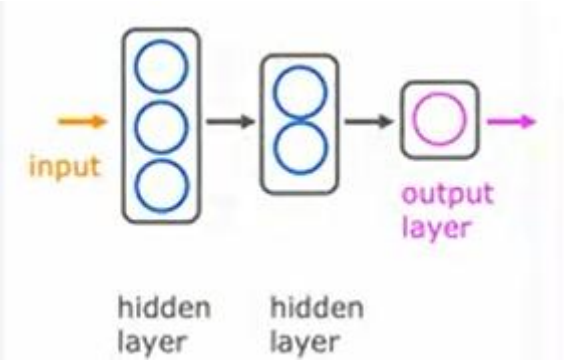
What you see here is that there is a logistic regression algorithm or logistic regression unit that is taking as input, affordability, awareness, and perceived quality of a t-shirt, and using these three features to estimate the probability of the t-shirt being a top seller. This is just logistic regression. But the cool thing about this is rather than using the original features, price, shipping cost, marketing, and so on, is using maybe better set of features, affordability, awareness, and perceived quality, that are hopefully more predictive of whether or not this t-shirt will be a top seller.



One way to think of this neural network is, just logistic regression. But as a version of logistic regression, they can learn its own features that makes it easier to make accurate predictions. In fact, you might remember from the previous course, this housing example where we said that if you want to predict the price of the house, you might take the frontage or the width of lots and multiply that by the depth of a lot to construct a more complex feature, x_1 times x_2, which was the size of the lawn.

There we were doing manual feature engineering where we had to look at the features x_1 and x_2 and decide by hand how to combine them together to come up with better features. What the neural network does is instead of you needing to manually engineer the features, it can learn, as you'll see later, its own features to make the learning problem easier for itself.

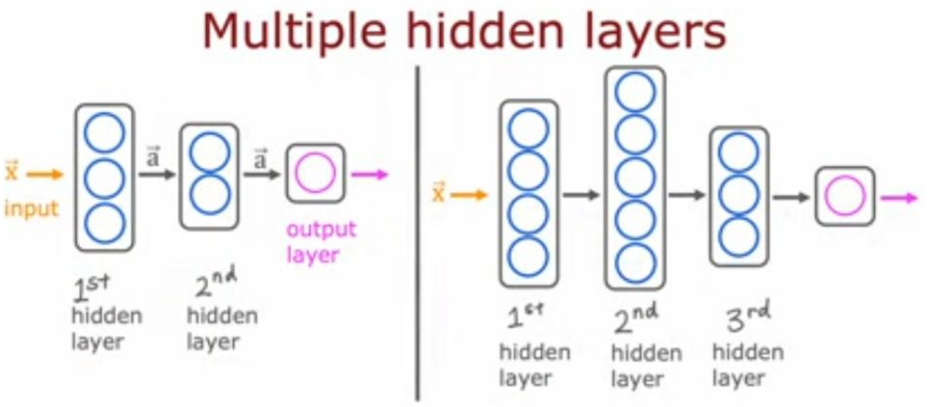
This is what makes neural networks one of the most powerful learning algorithms in the world today. To summarize, a neural network, does this, the input layer has a vector of features, four numbers in this example, it is input to the hidden layer, which outputs three numbers.



I'm going to use a vector to denote this vector of activations that this hidden layer outputs. Then the output layer takes its input to three numbers and outputs one number, which would be the final activation, or the final prediction of the neural network.

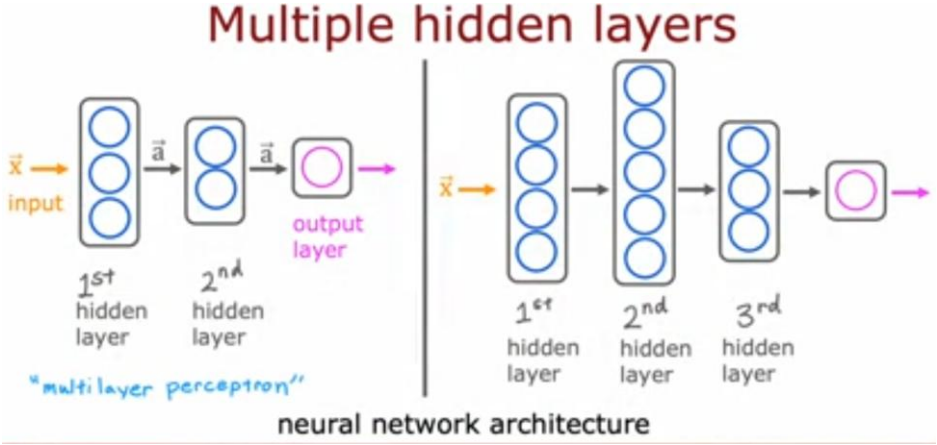
One note, even though I previously described this neural network as computing affordability, awareness, and perceived quality, one of the really nice properties of a neural network is when you train it from data, you don't need to go in to explicitly decide what other features, such as affordability and so on, that the neural network should compute instead or figure out all by itself what are the features it wants to use in this hidden layer.

That's what makes it such a powerful learning algorithm. You've seen here one example of a neural network and this neural network has a single layer that is a hidden layer. Let's take a look at some other examples of neural networks, specifically, examples with more than one hidden layer. Here's an example.



This neural network has an input feature vector X that is fed to one hidden layer. I'm going to call this the first hidden layer. If this hidden layer has three neurons, it will then output a vector of three activation values. These three numbers can then be input to the second hidden layer.

If the second hidden layer has two neurons to logistic units, then this second hidden there will output another vector of now two activation values that maybe goes to the output layer that then outputs the neural network's final prediction. Here's another example. Here's a neural network that it's input goes to the first hidden layer, the output of the first hidden layer goes to the second hidden layer, goes to the third hidden layer, and then finally to the output layer.



When you're building your own neural network, one of the decisions you need to make is how many hidden layers do you want and how many neurons do you want each hidden layer to have. This question of how many hidden layers and how many neurons per hidden layer is a question of the architecture of the neural network. You'll learn later in this course some tips for choosing an appropriate architecture for a neural network.

But choosing the right number of hidden layers and number of hidden units per layer can have an impact on the performance of a learning algorithm as well. Later in this course, you'll learn how to choose a good architecture for your neural network as well. By the way, in some of the literature, you see this type of neural network with multiple layers like this called a multilayer perceptron. If you see that, that just refers to a neural network that looks like what you're seeing here on the slide.