Deep Learning Intro.

- teaches a computer to filter inputs through layers to learn how to predict and classify information. - Observations can be in the form of images, text, or sound.
- attempts to mimic the activity in layers of neurons in the neocortex.
- You get input from observation and you put your input into one layer. That layer creates an output which in turn becomes the input for the next layer, and so on. This happens over and over until your final output signal!

- The input information is broken down into numbers and the bits of binary data that a computer can use. (We need to either standardize or normalize these variables so that they're within the same range.)

- What about synapses? Each of the synapses gets assigned weights.
- By adjusting the weights, the ANN decides to what extent signals get passed along.
- When you're training your network, you're deciding how the weights are adjusted.

How do ANNs run?

- In neural networks, you tell your network the inputs and what you want for the outputs, and let it learn on its own.
- We can avoid the necessity of entering in all the rules.

The information goes back, and the neural network begins to learn with the goal of minimizing the cost function by tweaking the weights. This process is called backpropagation.

In forward propagation, information is entered into the input layer and propagates forward through the network to get our output values. We compare the values to our expected results. Next, we calculate the errors and propagate the info backward.

- Backpropagation allows us to adjust all the weights simultaneously.

What is a weighted sum?

-Inputs to a neuron can either be features from a training set or outputs from the neurons of a previous layer.

- Each connection between two neurons has a unique synapse with a unique weight attached.
- If you want to get from one neuron to the next, you have to travel along the synapse and pay the “toll” (weight).
- The neuron then applies an **activation function** to the sum of the weighted inputs from each incoming synapse. It passes the result on to all the neurons in the next layer.
- When we talk about updating weights in a network, we’re talking about adjusting the weights on these synapses.

- A neuron’s input is the sum of weighted outputs from all the neurons in the previous layer.
- Each input is multiplied by the weight associated with the synapse connecting the input to the current neuron.

What is an activation function?

- **The activation function of a node defines the output of that node.**
- translates the input signals to output signals.
- maps the output values on a range like 0 to 1 or -1 to 1.
- a number that represents the likelihood that the cell will fire. At it’s simplest, the function is binary: yes (the neuron fires) or no (the neuron doesn’t fire).
- If you were using a function that maps a range between 0 and 1 to determine the likelihood that an image is a cat, for example, an output of 0.9 would show a 90% probability that your image is, in fact, a cat.

Four Common types of Activation Functions:

Threshold function:

- This is a step function. If the summed value of the input reaches a certain threshold the function passes on 0. If it’s equal to or more than zero, then it would pass on 1.
- It’s a very rigid, straightforward, yes or no function.

Sigmoid function:

- This function is used in logistic regression.
- Unlike the threshold function, it’s a smooth, gradual progression from 0 to 1.
- It’s very useful in the output layer and is heavily used for linear regression.

Hyperbolic Tangent Function:

- Unlike the sigmoid function which goes from 0 to 1, the value goes below zero, from -1 to 1.
- Neural networks sometimes get “stuck” during training with the sigmoid function. This happens when there’s a lot of strongly negative input that keeps the output near zero, which messes with the learning process.
- Example hyperbolic tangent function (tanh)

Rectifier function:

- It’s the most efficient and biologically plausible.
- Even though it has a kink, it’s smooth and gradual after the kink at 0. This means, for example, that your output would be either “no” or a percentage of “yes.”
- This function doesn’t require normalization or other complicated calculations.

The End.