Day 3: Deep Neural Networks—From Data to Architecture

Objectives:

(1) Understand the pipeline and importance of data to designing a deep neural network.
(2) Understand the concepts underlying a DNN
(3) Be able to import and use your own data to train a network

Lesson Plan:

Module 1: The importance of good data

There are two options we can take to take here.

One is to use the Iris Training exercise as illustrated on Tensorflow's website. This version is more engaging, and requires taking students to the local botanical garden to measure Iris features according to those used in the Iris dataset. These then become the test-data that we will use after building our Iris DNN.

The second is to continue using the fresh-start data set. This then becomes an example of how bad data can affect model outputs, as the accuracy should be approximately 35% after robust training.

15-minute break

Module 2: How does a DNN work??

Building an ML graph:
Begin by drawing the underlying equation for a DNN on the board: y=Wx+b. Explain what each of the variables mean. Now, show how this equation looks in real time . . . use matrices as drawn on the board to illustrate the matrix multiplication that belies the method. Do one where the number of units in the hidden layers is equal to the inputs, but then show the difference when you add more units to the hidden layer, or more inputs.

Activation Function:
Begin by showing how the weights are actually the connections between neurons in the graph (do this visually by illustrating how all the weights are actually the connections from one input to a particular neuron). What we actually train are numeric values for the weights in our network, so these are the things that change over time—not our inputs (x) or our outputs (y). The activation function decides whether the neuron that the weights connect to actually fires a 1 instead of remaining a zero for the next layer. (diagram sigmoid and Relu neuron activations).

Stochastic Gradient Descent and Back-Prop:
So does that make sense so far? Here's where we figure out HOW to train those weights. What we're attempting to do is find the minimum distance between each neuron. The shortest path between neurons is th one we want to pick, and to do that we use partial derivatives in a function called stochastic gradient descent. The weights are like like peaks and valleys on a map, and what we want is to find the lowest point in a particular valley. (either show this visually via a map on the screen, or do the human links exercise, where students create a human knot and then call out where they're low or high. This latter exercise is a sure-fire way to illustrate the function of stochastic gradient descent).

15-minute break

Module 3: Building a DNN

Have students open up a new file titled dnn.py in the Day-3 folder in their project folder. Make sure that all students have the requisite data files.

Walk students step-by step through of building the network, paying attention to the following key points:

-importing a file: The best bet here is to do this using Pandas. The reason being is because we want them to be able to use a litany of different data docs, and so using tensorflow's built in import functions is limiting students' outcomes in this regard.

-Creating an input function: describe to students that this creates a dictionary on the fly for the data that they're going to be using. Show them step by step the building of the input function.

-Converting strings to text using .layers: include a discussion why we need to convert a string to a numerical value.

-Embedding layers as dnn inputs . . . seriously. TF lives off of tensors, so we should convert these things to tensors whenever possible. Here is a good point to talk about how strings are converted to tensors using CBOW in tf if time permits. This concept is the first stepping stone for most people into NLP, making the research area accessible.

-Setting up the model.

-training.

Conclusion & Individual questions period.