Assignment 3: Introduction to Neural Networks

1. **Feed Forward Neural network for Binary XOR:** In this task you will implement and understand a simple single hidden layer neural network with 2 neurons in the hidden layer for the binary xor problem.

   A. Inspect starter.py and understand the code. Run the model and plot the loss on a loglog plot. Does the currently implemented trained model successfully implement this problem? Hint: the loss should be *very* close to 0. You may also want to inspect the output values.

   B. One way to better understand what our model is doing is by running the model on a grid of values beyond where it was originally trained. Write a function that returns a grid in a box defined on the intervals [x=[-0.5,1.5] , y=[-0.5,1.5]] consisting of 100^2 points. Make sure the grid is flattened in each dimension so that we can run it through our model. *(Hint: numpy.linspace, numpy.meshgrid)*. Run the data through the model and plot a scatter plot using the x,y coordinates defined above and the output of the model as the color. In addition .plot the 4 points ((0,0), (0,1), (1,0), (1,1)) and color code by their correct label.

   C. From the scatter plot it should be clear that there is at least one separating line in input space that is being used to classify the input. Write out the equation for one of the lines. (Hint: w_0 x + w_1 y = -b, but currently our b=0). Now write a function called *make_explicit_line_equation that* takes weights (w) and biases (b) as input and returns a new function of x in explicit form. Add the line to the scatter plot created in part 1.B. Set the limits of the plot to something reasonable, say xlim=(-1,1.1), ylim=(-1.1.1).

   D. Similarly we can visualize the second layer by plotting a scatter plot on our grid using the intermediate output (y1) from our model. Make a scatter plot similar to 1.C both before and after applying the sigmoid,

and plot the separating line after the sigmoid. Make sure to visualize the original four training points in each space and color code by their correct label.

E. (Graduates Only) Replace the sigmoid activation in the first hidden layer with a ReLU, and repeat 1.CD, does the model learn different hyperplanes (separating lines)? Which activation function produces separating lines do you think are closer to the correct answer?

F. From the above experiments it is now clear that the first (hidden) layer defines separating hyperplanes to slice up the input space, while the second (output) layer combines these spaces into the correct label. The issue with our model is therefore that since our bias is 0, all of our separating lines must be drawn through the origin. Add a bias parameter (nn.Parameter) to both layers in our network. And visualize the results as done in 1.ABCD.

2. **Generalized XOR:**

A. Write a new function called generalized_xor that takes in a d-dimensional binary vector, $b = b_d b_{d-1} \ldots b_1$ and outputs a cascading xor, $o = \text{XOR}\left(b_d, \ldots \text{XOR}(b_3, \left(\text{XOR}(b_2, b_1)\right)\right)$, where b_d is the most significant bit. Make sure it's a vectorized implementation. Create a new binary training set B, by calling make_binary_arrays for d=3, and apply your function to get the generalized xor output.

B. Make a 3d scatter plot of the data defined in part 2.A using any software of your choice. Take a screenshot or save a figure of the visualization. Based on visualizing the data how many planes do you think are needed to separate the data?

C. Verify your guess by training a model with k=[2,3,4,5,6,7,8] number of planes, make a semiology plot of k vs. the minimum loss for each k (you can just take the loss at end of training). At which k does the model start to correctly classify the data.

D. (Graduates Only) Add the minimum number of planes to your visualization in part 2.B and inspect that they do indeed separate the data.

E. Repeat parts 2.AC, for d=4,5,6. Based on your results how many hyperplanes does it take to successfully separate the d-dimensional xor problem?