Problem Set 2: Neural Networks
IMT 575: Scaling, Applications, and Ethics

Instructor: Lovenoor (Lavi) Aulck
University of Washington
Spring Quarter 2020

April 13, 2020

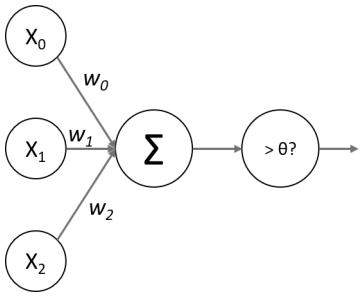
Instructions

All problem sets must be submitted as a Jupyter notebook. You must organize your analysis and utilize the notebook's markdown capabilities to present your solutions. Partial credit will be awarded for each question for which a serious attempt at finding an answer has been shown. Please see the course syllabus for more instructions/information. A rubric for this assignment is available on the Assignment's canvas page. Unless otherwise noted, this assignment is due on **April 27, 2020.**

Part 1: Perceptrons by hand

You can perform the calculations for Part 1 and create the plots using python or by hand. If by hand, insert images of your work (one per question) into your Jupyter notebook and include the image file(s) with your submission to Canvas.

For Q1-4, use the perceptron shown below.



1. Suppose you had a perceptron with the inputs and outputs shown below (corresponding to an OR function), where y is an output and X_i is an input.

X_1	X_2	y
3	1	1
0	1	1
3	0	1
0	0	0

   Give weights to the perceptron to compute this function. Assume X_0 is a bias term = 1 and θ = 0. What are the weights? Plotting X_1 vs X_2 and using shapes/colors to indicate y values, show the separation line for the perceptron with the weights you gave.

2. The separation line you showed in Q1 is one of many possible separation lines. Suppose w_0 = -1 with X_0 and θ holding the same values as Q1. Given the same inputs/outputs from Q1, what are the bounds on w_1 and w_2? (Phrased differently: are there minimum/maximum values for w_1 and/or w_2? If so, what are they?)

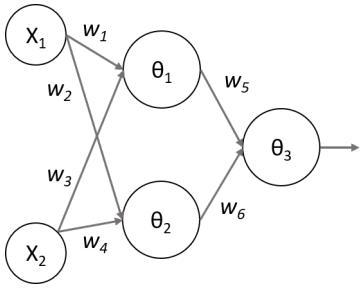
3. Suppose you had a perceptron with the inputs and outputs shown below (corresponding to an AND function), where y is an output and X_i is an input.

X_1	X_2	y
2	5	1
2	0	0
0	5	0
0	0	0

   Give weights to the perceptron to compute this function. Assume X_0 is a bias term = 1 and θ = 0. What are the weights? Plotting X_1 vs X_2 and using shapes/colors to indicate y values, show the separation line for the perceptron with the weights you gave.

4. The separation line you showed in Q3 is one of many possible separation lines. Suppose w_0 = -1 with X_0 and θ holding the same values as Q3. Given the same inputs/outputs from Q3, what are the bounds on w_1 and w_2? (Phrased differently: are there minimum/maximum values for w_1 and/or w_2? If so, what are they?)

5. Use the multi-layer perceptron shown below. Note that we will not include a bias weight/input for this perceptron. The inputs are densely connected to the first layer, and though not indicated, all inputs must be multiplied by the appropriate weights and summed before comparing to activation thresholds. All nodes will output a 1 when activated and a 0 when inactive. θ values indicate activation thresholds for a particular node and inputs to the function are indicated with X_i values. Suppose you had the inputs and outputs shown below (corresponding to an XOR function), where y is an output and X_i is an input.



X_1	X_2	y
1	1	0
1	0	1
0	1	1
0	0	0

   Suppose you were given the following: $w_1 = w_2 = w_3 = w_4 = 1$. Set values for w_5, w_6, θ_1, θ_2, and θ_3 to compute the function. Show that the values you selected work in computing this function.

Part 2: Rosenblatt's Algorithm

6. Import the Boston housing dataset using sklearn.datasets. Create a new column called "highPriced" that is a boolean indicator of whether the target value in the dataset (i.e. MEDV or median home value) is greater than \$40K (note that this is a different threshold than PS1). Show that this variable is linearly separable across the MEDV and RM variables by plotting in a two dimensional space. Comment on what you see.

7. We will use the RM and MEDV variables to implement Rosenblatt's algorithm. Start by creating an X input matrix and a Y output vector. X should have dimensions 506x2 (i.e. 506 observations and 2 columns - RM and MEDV). Y should have dimensions 1x506 (i.e. the highPriced feature with 0/1 indications of whether an observation is high priced). Show the shapes of X and Y. (Note: It is recommended that you use numpy.ndarrays to store values. You can get the numpy.ndarray representation of a pandas dataframe using the .values attribute)

8. Next, add a column of ones to be the first column of X. This is our bias input. Now, X should be a 506x3 matrix with three columns: the bias, RM, and MEDV. Show that this is indeed the case by displaying the first 5 rows of X.

9. Now, set a step size that is fairly small (if your algorithm is taking too long to converge, you will need to adjust this value). Also, initialize a vector of weights that are between 0-1. Note that you should have one weight per column (Note: it is recommended to store your weights in a numpy.ndarray). Show your step size and starting weights.

10. Create a function called myPerceptron that takes in the following four inputs: X, Y, a vector of weights, and your step size. The function should pass through a single iteration of Rosenblatt's algorithm - we will run this function until convergence in the next question. The function should output two things every time it is called: the vector of updated weights and a count of the number of misclassifications. (Note: it is highly recommended that you use dot products to multiply the weight vector with the input matrix in calculations. Also keep in mind that $\hat{y}$ are your predictions and must be thresholded to 0 or 1 when working through the algorithm.)

11. Run your myPerceptron function using your starting weights and step size until convergence. One possible structure for this is to wrap it in a while loop (a snippet of this setup is shown below). Track the number of iterations it takes to reach convergence and the errors after each iteration as outputted by your function. If your algorithm takes more than 40K iterations to converge, go back and adjust the step size. Once finished, plot the number of errors for each iteration vs the iteration number. How many iterations did the function take to converge? What does the shape of the plot of the errors look like? We discussed gradient descent for multi-layered networks in the lecture videos - do you think gradient descent would generate a plot of errors vs iteration (epoch) that looks similar? Why or why not?

```
(Initialize inputs here)
errors = 1
while errors > 0:
    weights, errors = myPerceptron(X, y, weights, step)
```

12. The values of your weights after convergence can be used to draw a separation line where: $w_0X_0 + w_1X_1 + w_2X_2 = 0$

Consider your plot from Q6. We want to superimpose the separation line that we calculated onto the plot. Solve the equation above for whichever variable is on the y-axis of the plot from Q6, be it X_1 or X_2. In other words, solve the equation above such that you have something of the form $y = mx + b$, where y is the variable on the y-axis of the plot from Q6 and x is the variable on the x-axis. Remember that X_0 is our bias term of 1s and $w_0 X_0$ is thus a constant. Plot this line onto a copy of your plot from Q6 and comment on what you see.

Part 3: Out-of-the-box Neural Networks

14. Create a training/test split for the entirety of the original Boston housing dataset (highPriced excluded but MEDV included). The MEDV variable will be our output (target variable) going forward; all other features/variables will be our input variables. Use 80% of observations for the training set and 20% for the test set. What are the shapes of your training and test datasets? (Hint: you may want to consider looking into sklearn.model_selection's train_test_split function)

15. First, we're going to train multiple neural networks, each with a single hidden layer, on our training dataset. Using sklearn's MLPRegressor, train a neural network with a single hidden layer of each of the following sizes: 2, 5, 10, 20, 50. Set the maximum number of iterations to 1000 while keeping all other parameters as defaults. How well does each neural network perform when predicting values for the training dataset? How well does each neural network perform when predicting values for the test dataset? Comment on what you find. (Note: you can iterate over this process in a for loop while tracking how well each network performs)

16. Next, we're going to train multiple neural networks, each with two hidden layers, on our training dataset. Using sklearn's MLPRegressor, train a neural network with hidden layer sizes: (2, 2), (5, 5), (10, 10), (20, 20), (50, 50). Set the maximum number of iterations to 1000 while keeping all other parameters as defaults. How well does each neural network perform when predicting values for the training dataset? How well does each neural network perform when predicting values for the test dataset? Comment on what you find. (Note: you can iterate over this process in a for loop while tracking how well each network performs)

17. Using sklearn's MLPRegressor, design a neural network architecture that you believe gives you the best performance on the test dataset. Change only the hidden layer sizes (have as many layers of whatever size you'd like) and set the maximum iterations as you see fit. Keep other parameters to be defaults. What is the architecture you designed? Describe your process for developing the design and the design's performance relative to the networks you built in Q14 and Q15. (Note: you will not be graded on your ability to find the *absolute* best architecture; rather, you will be graded on your process for designing an architecture that has strong performance and your reasoning behind the process.)