**CSCE 823: Machine Learning**

**Summer 2019**

**HW1**

Due Wed, 3 July 2019 at 2359

Submit via Canvas

**(**This Homework is worth 10 points toward your final grade**)**

In this assignment, you will explore using perceptrons for classification. You will be starting with the provided jupyter notebook code in this assignment. You will be evaluated on your applications of techniques and methodology, as well as the evidence you present and conclusions you draw with respect to the models.

Your homework will be composed of an integrated code and report product using Jupyter Notebook. In your answers to written questions, even if the question asks for a single number or other form of short answer (such as yes/no or which is better: a or b) you must provide supporting information for your answer to obtain full credit. Use python to perform calculations or mathematical transformations or generate graphs and figures or other evidence that explain how you determined the answer. Each step listed below should correspond to code and/or markdown in your report file. Use numbered comments in your code and numbered text segments (headers) in markdown to help identify the location of your answer.

This assignment uses Keras (with TensorFlow backend) to model networks quickly. Be sure to read through the Keras documentation and understand the different options from which you choose. Time spent understanding Keras now will help in future assignments and your project. You will not need the GPU backend working for this assignment.

Each step listed below should correspond to code and/or text in your file. Make it easy for the instructor to find your work by using the step identifiers (for example: “**Step 1:**”) clearly identified in both your code and your writeup.

**Simple Perceptron (code provided)**

In steps 1-8 keras code is provided which uses a simple perceptron to solve several different datasets including logic gates and the NUT database from Fyfe’s paper. Run (and rerun) this code with different learning rates & maximum epochs of training, look at the results and make observations. Because these datasets are extremely small, we are reusing the training set as the validation/test set – which is normally a violation of the golden rule.

Notice that in step 8 the simple perceptron fails to learn the XOR function.

**Multilayer Perceptron:**

In step 9 your job is to build a multilayer perceptron (using keras) which can, with high probability, consistently learn the XOR function using as few layers and as few neurons per layer as possible. Can you achieve the goal using only 2 neurons in the first layer and 1 neuron in a second layer?

Step 9: Build a *new* keras model with multiple perceptrons to handle the XOR problem. This model should have 2 inputs and one (Boolean) output (hard\_sigmoid activation) but *will need more than one layer* to solve XOR. Report performance of a model on the XOR dataset - in substeps 9a through 9c (below). You should consider using different model architectures (number of layers, layer widths, inner-layer activation functions), optimizers, and learning rates and see how these choices affect the model fitting process and performance (perhaps determined empirically inside a cross-validation loop). Your goal is to achieve 100% accuracy on XOR. How well can you do? What is the smallest number of layers and nodes per layer you can achieve success with?

* 1. Instantiate a model in keras and fit the model on the training data. Use a batch\_size of 1. You will need to decide on the number of epochs to train for (and select an appropriate learning rate). Alternately, you can use the keras EarlyStopping callback to stop once you’ve obtained 100% accuracy. Capture the model history for reporting in a later step.
  2. Use keras model.evaluate, to obtain and display the score of your model on the full training dataset (batch\_size = 4). Display graphical plots of the model history over epochs (training & validation accuracy vs epoch; loss vs epoch).   
     Repeat the training/evaluation process under different conditions to see how quickly or slowly your model converges and present performance graphs.   
     Explore different layer widths, number of layers, activation functions, learning rates, optimizers.  **Discuss the performance as a function of your design choices**.
  3. Use keras get\_weights() on the model architecture and design choices that worked the best to obtain & report the final weights of the perceptrons. Using the instructor-provided code, generate the final decision boundary. Describe the shape of the decision boundary. **Describe how the weights of the perceptrons relate to this decision boundary.**

**Optional/Ungraded Challenge:** Make a movie of the decision boundary changing over time as XOR is learned by your network. This movie should contain 1 image frame per epoch.

**Rules of Engagement for this Homework Assignment:**

**Using external sources:**

The use of pre-existing solutions to answer assignments is not allowed. This includes the use of other students’ answers, answers found on the internet, solution manuals, and any other source of information which does not reflect your own work.

You may use the internet or get help from peers when determining basic things like “how do I add points to a plot in Python”, but don’t try to search for specific answers to problems I ask in the homework.

You may use any pseudocode or concepts learned in class to solve the problem.

The code you write must be original work unless otherwise specified by the instructor.

**Programming Conventions**

In the code chunks, good software engineering principles apply: self-documenting code (meaningful function & variable names), additional comments and whitespace should be the standard for all code you turn in. If your code is not understandable it may not receive full credit.

IMPORTANT: You should explain what you are doing in text as well as in the comments to a code chunks. A rule of thumb is to have line-level comments in the code chunks and save the larger high level comments & observations for the text in your homework writeup. Make mention of the names of your python files in your writeup so it is clear how you are using them

**Code File Structure and Naming Conventions**

Rename your homework jupyter notebook file with the name: “LASTNAME\_FIRSTNAME\_HW1.ipynb” . If you create additional output files (like movies of the changing decision surface), ensure that your code & data is contained entirely in one directory, then zip & name the file “LASTNAME\_FIRSTNAME\_HW1.zip” before submitting.

**Pre-submission Checklist:**

**Code/Data:** Ensure all of your code and data is located in a single directory (you can put your document in the same directory). Ensure that your code runs when python is set to make that directory your working directory. Avoid using absolute pathnames when loading data in scripts as your pathname will not be the same as the instructor’s pathname.

**Document:** Make sure you proofread your document carefully to ensure your final product reflects what you intend to submit. You don’t need a separate title/name page, but make sure your name is easy to find in the text of the document (ex: first line, or in the header). Include page numbering on your document. Indicate the submission date on your document Use page numbers.

**What to Submit:** Submit the single jupyter notebook (or your **zip** file of the folder containing all of your products) to Canvas. Note that if you discover an error and change a problem solution and re-submit, keep in mind that your instructor will only review your *latest* submission on Canvas – make sure it is complete.