Homework #2

COMPSCI 591/691 NR

This is a programming homework. All programming assignments in this course will be using Python 3. For this assignment you are supposed to submit a Jupyter notebook (https://jupyter.org/install)

The format of the notebook should be in form of a research notebook. Use both the text block to describe/ explain your process.

In this assignment, we tackle the problem of double half moons, as described below. Consider the following toy dataset:

![Diagram of double half moons]

**FIGURE 1.8** The double-moon classification problem.
Consider a dataset with 2000 data points. Train a perceptron to classify the two areas. Report the following:

1. Create a function to generate data points given the w, d and r. You may use numpy for this part. For help on this part, refer to the `make_moons` of scikit learn. [https://github.com/scikit-learn/scikit-learn/blob/b194674c4/sklearn/datasets/_samples_generator.py#L649](https://github.com/scikit-learn/scikit-learn/blob/b194674c4/sklearn/datasets/_samples_generator.py#L649)
2. Display a dataset for r=10, w=6 and d = 1 with 2000 data points. 1000 in region A and 1000 in Region B
3. Train a Rosenblatt perceptron using perceptron learning on the dataset as described in Bishop textbook, Sec 4.1.7 (pp. 192-197).
4. Display the training curve and the Mean squared error over the training and plot the decision boundary.
5. Create a test dataset with 4000 data points and report the test accuracy.
6. Repeat steps 2-4 with d=0. Comment on the differences.
7. Implement a batched version of the perceptron and comment on how it performs with respect to the single sample correction on d=0.

You are now tasked with creating a neural network for an artificial entity whose survival depends on correctly identifying the two regions. However, the two regions may not always be linearly separable. But to combat this, you are given a budget of x neurons. Implement a simple neural network with any non-linearity you like. For this part of use d=-2. Do not use libraries like Pytorch or Tensorflow. Implement the back-propagation yourself. Do not use the test dataset for parameter search. Design the best neural network you can to classify the dataset for x = 3, 5, 10, 50. Plot the learning curves, test accuracy and the decision boundaries. Comment on the performance. You may use weight regularization and batch updates.