Neural Networks Lab Manual 3/4

From the Single Unit to MNIST — A Process-Over-Product Sequence

---

Lab 3 — From a Straight Line to a Hidden Layer (25 pts)

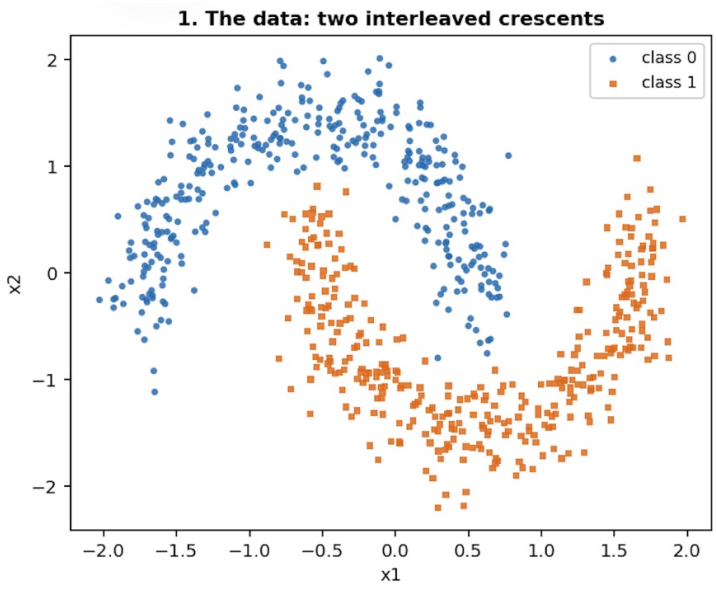
Idea

Lab 2 showed that a hidden layer makes XOR learnable, on four points. This lab shows the same thing on real data you can look at, and then asks what the network actually learned. Two experiments: what a linear model can do and what the hidden layer adds. Every experiment follows the same skeleton: **predict → run → visualize → reconcile.**

Dataset — two-moons (provided)

Use the two files supplied with this manual: `two_moons_train.csv` (700 points) and `two_moons_val.csv` (300 points). Columns are `x1, x2, y`.

About the data: two interleaved crescents, two input features, binary labels, roughly balanced. It is not linearly separable, so it needs the hidden layer you earned in Lab 2.



Scaffolding

The Lab3.zip file comes with two files that are designed to read in the data and produce the plots: MoonsData.java and Plot.java. You should not touch them. You need to provide two classes, one a perceptron the other a feed-forward network with a hidden layer that can have an arbitrary size. In both cases, the input is two data points and the output is one neuron. I am making available Lab3.java, in which I interact with PerceptronBias.java and MLPBias.java, my own solutions. The Lab3.java file should give you a sense of how you want to interact with those two models. Notice that both of those classes implement the provided Model.java interface. Feel free to have AI translate the code to Python. You may not use NumPy.

Experiment 3.1 — What a straight line can do

Train a model with **no hidden layer** and a sigmoid activation function on two-moons. Before you run it, predict the accuracy it will reach, and sketch where you think the boundary will sit. Accuracy is the percentage of correctly classified datapoints. You may wish to reuse the single layer perceptron with a sigmoid activation function from lab 1. Plot the boundary with misclassified points highlighted.

Experiment 3.2 — What the hidden layer buys

Create a new class, possibly by copying and pasting. Keep the sigmoid output unit and add one hidden layer of ReLU units and train again. Predict the accuracy first and predict what shape the boundary will take. Use sigmoid at the output layer.

Experiment with several hidden layer sizes and see how that affects accuracy. Report accuracy as well as the "number of kinks in the line." Report the minimum number of hidden units that has good enough prediction. Explain what you mean by "good enough."

Experiment 3.3 — Three ways to prevent training.

Taking the code for 3.2, identify and test three different and separate ways in which to modify the code so that the network does not learn or does very poorly. (With a lit bit of luck, you encountered those issues while developing your code.) The more subtle the better.

Deliverables and points

Item	Artifact	Pts
Predictions for 3.1–3.2	Prediction ledger	3
3.1: result: accuracy data and visualization. Explanation of why no straight line can separate the crescents, and where the errors fall.	Mechanism explanation, Visualization	5
3.2: result: accuracy data and visualization. Explanation of the effect of the number of layers on separability of the crescents.	Mechanism explanation, Visualization	8
3.3: Description of three cases	Mechanism explanation	6
Reconciliations across 3.1–3.2	Reconciliation	3