Assignment 3 – Artificial Neural Network

Due date: Saturday December 3 at midnight.

Objective

Using the MNIST handwritten digits dataset, build an Artificial Neural Network (ANN) classifier that classifies hand-written images of 0-9 digits. **You must code the BACKPROPAGATION algorithm from scratch as described in the book** (page 420 in *Kelleher et al*). **You cannot use ready-made open source or commercial Machine Learning packages such as TensorFlow, PyTorch, etc**.

The Training and Testing Data

Enclosed in the "data" subfolder are 4 files:

- training60000.csv: 60,000 training images.
- training60000_labels.csv: Labels of the above 60,000 training images.
- testing10000.csv: 10,000 testing images.
- testing10000_labels.csv: Labels of the above 10,000 testing images.

The data has been normalized using Range Normalization according to the following formula with high=1 and low=0.01.

$$a_i' = \frac{a_i - min(a)}{max(a) - min(a)} \times (high - low) + low$$

Network Topology

Your ANN topology should have:

- 1 INPUT layer with 784 inputs (sensing neurons)
- 1 HIDDEN layer with X logistic units. Or 2 HIDDEN layers with X and Y logistic units, respectively. I suggest you try both and use the most accurate one.
- 1 OUTER layer with 10 softmax units.

For example, if you use 1 HIDDEN layer, your network topology should look somewhat like Figure 1. For a 2 HIDDEN layers network, you will use another hidden layer with Y logistic units

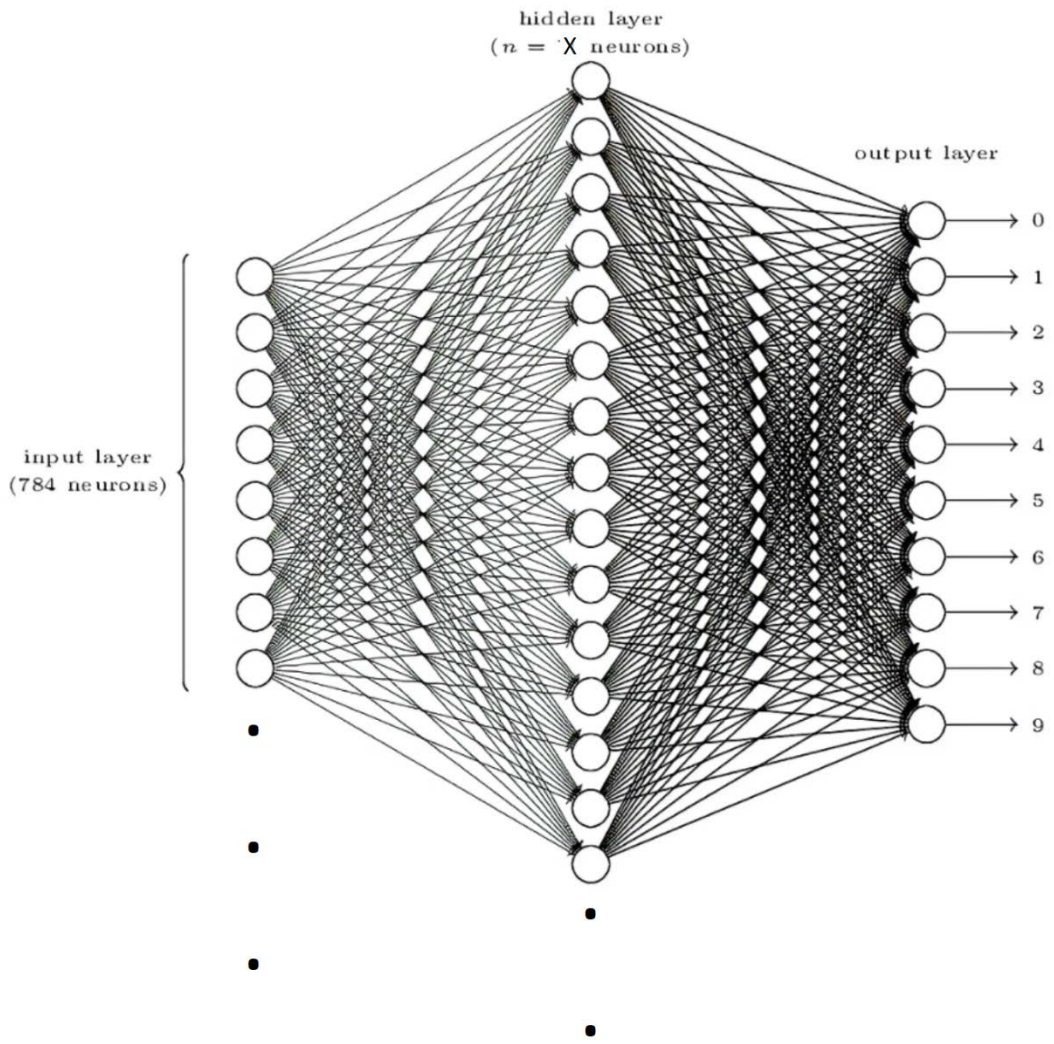

Figure 1: Network topology when 1 HIDDEN layer is used.

- The 784 inputs correspond to the 784 pixels of each image (28 x 28 = 784). The pixels of each image were flattened into a 1D array using row-major order.
- The 10 output nodes correspond to the ten digits (0 to 9) that the network is going to learn to predict.

Activation Functions to Use

Use the *logistic* function for the HIDDEN layer(s). The *logistic* function is defined as:

$$f(z) = \frac{1}{1+ e^{-z}}$$

Use the *softmax* function for the OUTER layer. The *softmax* function is defined as:

$$\varphi_{sm}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{m} e^{z_m}}$$

The softmax function squashes the Z values in the OUTPUT layer to values between 0 and 1 that can be interpreted as probabilities. **Read about the use of the *softmax* function on pages 463-464 in *Kelleher et al*. And look at Figure 8.27 on page 465.**

Calculating δ's

The **cross-entropy** error function is normally used in situations where the output of the output layer neurons are interpreted as a probability distribution. **Read about the use of the *cross-entropy* function on pages 465-469 in *Kelleher et al*.**

For the output layer:

$$\boldsymbol{\delta}_{k=\star} = -\left(1 - \hat{\mathbf{P}}_k\right)$$

$$\boldsymbol{\delta}_{k\neq\star} = \hat{\mathbf{P}}_k$$

That is, for the output neuron that correspond to the one-hot encoded target label, $\delta = -(1 - P_k)$.
For all other output neurons, $\delta = P_k$ (that is, their *softmax* activation value). **See equations 8.80 and 8.81 on page 469 of *Kelleher et al*.**

For inner layer(s):

$$\delta_k = \underbrace{(logistic(z) \times (1 - logistic(z)))}_{} \times \left(\sum_{i=1}^{n} w_{i,k} \times \boldsymbol{\delta}_i\right)$$

Testing Your Model

When done with training, test your model with the test datasets (testing10000.csv and testing10000_labels.csv). Print out the network properties and the classification accuracy (something that resembles the below):

```
==== Results
Network properties:  Input: 784, Hidden: X, Output: 10

Correct classification:  …
Incorrect classification:  …
Accuracy:  … %
```

Where Accuracy = (Correct / (Correct + Incorrect)) x 100

Remember that you are classifying 10 different digits. Without a classifier, guessing gives you around 10% accuracy. If you get >= 90% accuracy, you are effectively doing 9 times better.

What to Submit

- Source code of your ANN classifier (in Python file or a Jupyter Lab notebook if you used Jupyter Lab).

- Write a 1-page document (and convert to PDF) with the following:

    - 1 paragraph that describes the topology of your network.
    - 1 paragraph that describes the parameters you used for training (epoch, learning rate).
    - A snapshot of your testing output (similar to what is shown in blue above).
    - If you couldn't complete the algorithm, briefly describe where you got stuck and the errors you obtained.
    - (Optional): if you did some experimentations, describe what you did and what you observed. For example, if you tried different numbers of nodes in the HIDDEN layer include the results of all the numbers you tried. And what did you observe? If you tried 2 HIDDEN layers instead of just one, did it improve accuracy when compared to one layer?

- Create a folder and name it firstName_lastName_assignment3 (e.g. john_smith_assignment3)
- Put in the folder your source code (*.py file(s) or Jupyter Lab folder) and the 1-page PDF document.
- Zip the folder to generate file john_smith_assignment3.zip.
- Upload the zip file to Canvas before the due date (Friday December 10 at 11:59 PM).

Grading Rubric

- 50%: You have implemented the BACKPROPAGATION algorithm. Your code is well documented and easy to understand/follow. And you have encapsulated certain functionalities into re-usable functions.
- 35%: You obtained an accuracy >= 85%.
- 15%: You submitted the 1-page PDF document with the expected content.