Project Proposal

The System/Model

For my final project I will create a Hopfield Network that will classify handwritten characters, like digits, to their respective values. The dataset used is MNIST (Modified National Institute of Standards and Technology database). The MNIST dataset is quite large, containing 60,000 training images, and 10,000 testing images. The data has been size normalized and centered in a fixed-size image. Furthermore, the images are black and white, where white pixels represent the background, and black pixels make up the character/digit.

Description

A neural network is a mathematical model used to classify data based on how the neurons respond to the input. Neural networks are inspired by the structure and function of biological neural networks found in animals and humans. Hopfield Networks are recurent neural networks. They update their activation values asynchronously, based on the weights of surrounding connected neurons. This can be mathematically described as:

$$s_i \leftarrow \begin{cases} +1 & \text{if } \sum_j w_{ij} s_j \geq \theta_i, \\ -1 & \text{otherwise.} \end{cases}$$

OR

$$o = \left\{ \begin{array}{l} 1 : \sum_i w_i x_i \geq 0 \\ -1 : \sum_i w_i x_i < 0 \end{array} \right\}$$

where*:*

- *w_{ij} is the strength of the connection from neuron 'i' to 'j'*
- *x_i or s_j are the states of neuron 'i'*
- *θ_i is the threshold of neuron 'i'*

Furthermore, a Hopfield network must follow two important rules:

1. *Symmetry*
2. *No Self-connection*

Mathematically, the following property has to hold:

$$w_{ij} = w_{ji} \text{ and } w_{ii} = 0$$

The neurons are represented as a (weighted) matrix and based off the input data, updates are performed to the matrix. Using the equations above, overtime, the weights change and the matrix learns the input and is able to recognize patterns. For instance, it will learn what the letter 'A' looks like based off the spread and distribution of black pixels with respect to white pixels.

Solution

The Hopfield Network will be developed using Hebbian training. This will allow it to get better overtime and be able to classify objects not in the training set. For Hopfield networks, Hebbian learning is implemented in the following manner:

$$w_{ij} = \frac{1}{n} \sum_{\mu=1}^{n} \epsilon_i^\mu \epsilon_j^\mu$$

If the neurons 'i' and 'j' are equal, then the product will be positive, and have a positive effect on the weight 'w_{ij}'. The abstract idea behind this is, “Neurons that fire together, wire together. And neurons that fire out of sync, fail to link”.

Testing

1. Verification

The MNIST dataset contains 10,000 testing images. A subset of these images will be fed to the Hopfield network and the output generated will be observed and compared against the correct answer. This process will be repeated many times in order to verify that the neural network is functioning as intended. The images tested will be easy at first, and then gradually get harder.

Images that are difficult to classify can be tested by comparing the answer of the Hopfield network to answers generated by other (types) of neural networks.

2. Validation

The best way to validate the model is to have the user supply input data. The user will handwrite a character on a pop-up GUI, and specify the character in text, and the model will generate its answer. The two answers can then be compared to validate the model. This is the best way to test the model because it does not rely on the MNIST database, rather it checks realistic data generated by a real person.

The GUI can be created with Python's tkinter library, Java's Swing library, or QT for C++.

Exploration

The model can be tested on alternate data such as different fonts/typography. For example, Serif Vs. Sans-Serif fonts. This may cause unintended results, requiring the neural network to be (re)trained.

Potential Problems

1. **Training a neural net could be slow since:**
    - The CPU will be used to train the neural net, and not the GPU. Conventionally, GPUs are used to train large neural nets due to their ability to effectively perform large matrix multiplications. GPUs have more cores than CPUs, allowing them to process more numbers.
    - The process will not be effectively parallelized, training a neural network might be time consuming on one core and
    - In a Hopfield Network, since all nodes are inputs and outputs, and everything is fully interconnected, I will be dealing with N^2 weights
2. **Model is too difficult to implement**
3. **Unexpected results and difficult to debug**

References

- https://en.wikipedia.org/wiki/Hopfield_network
- http://web.cs.ucla.edu/~rosen/161/notes/hopfield.html
- https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_hopfield.htm
- http://perso.ens-lyon.fr/eric.thierry/Graphes2010/alice-julien-laferriere.pdf
- https://pmatigakis.wordpress.com/2014/01/18/character-recognition-using-hopfield-networks/
- https://en.wikipedia.org/wiki/Hebbian_theory
- https://medium.com/datadriveninvestor/what-is-hebbian-learning-3a027e8e4bbb
- https://en.wikipedia.org/wiki/MNIST_database
- http://yann.lecun.com/exdb/mnist/