The Perceptron Neural Network

Blake Moss

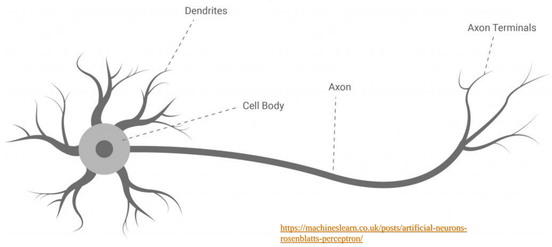
GitHub repository: https://github.com/blakemoss/PHYS-250-Project

Introduction

The Perceptron neural network is a tool which can be developed to accurately classify binary, linearly separable data. By making predictions and altering "weights" according to its error from the real result, the neural network can reach astounding prediction accuracies given enough time. Neural networks are universal approximators, and so their applicability is functionally infinite- this is why I decided to examine them for my final project. Instead of tackling a specific physics-based problem, I decided to use the computational methods developed in this course to create a single-layer Perceptron essentially from scratch. Although this approach limits the applicability of my project to linearly separable datasets, it also allows me to build a neural network from the ground up and deeply understand its mechanisms as opposed to using a pre-coded package like Keras and TensorFlow.
In this project I apply the neural network to both a generated dataset and a real-world dataset to show the power and versatility of this mathematical structure.

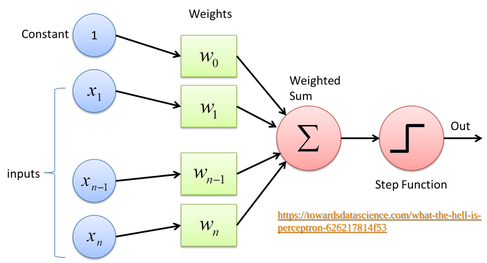
Introduction

The perceptron neural network, invented by psychologist Frank Rosenblatt in 1957, is designed to function as a simple neuron in the brain functions: by taking various inputs, applying some function, and generating an output.



https://machineslearn.co.uk/posts/artificial-neurons-rosenblatts-perceptron/

Below is a diagram showing the basic structure of the Perceptron neural network, which can be visually compared to the neuron pictured above.



https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53

Theory and Development

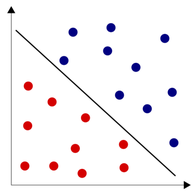
To make a prediction, the Perceptron multiplies each input of a datapoint by a weight, and applies a step function to make a prediction of the data point's classification. The weighted sum as depicted above can be calculated as a dot product between the input matrix and the weight matrix, with a bias added to improve accuracy:

$$\text{output} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} * \begin{pmatrix} w_1 & w_2 & \cdots & w_n \end{pmatrix} = (X \cdot W) + w_0$$

As the network is distinguishing between two classifications, a step function is appropriate: in this case, if the output is greater than 0, the function returns 1; else it returns 0.

$$\text{activation function} = \begin{cases} 1 & output \geq 0 \\ 0 & output < 0 \end{cases}$$

A key determiner of the applicability of a Perceptron neural network on a dataset is its ability to be linearly separated, as illustrated below:



https://commons.wikimedia.org/wiki/File:Separability_YES.svg

A single-layer Perceptron will not be able to correctly train to and make predictions on a dataset which is not linearly separable- this is because it functions by generating a decision boundary to separate the two groupings of data, as seen by the line in the picture above- for 3D data, this line would become a plane, and so on.

The neural network learns by making predictions on a dataset with known classification. For each datapoint, it makes a prediction of its classification and uses the error between its prediction and its actual grouping to update the weights which generate the next prediction. This process is called backpropagation, and is described in mathematical form below.

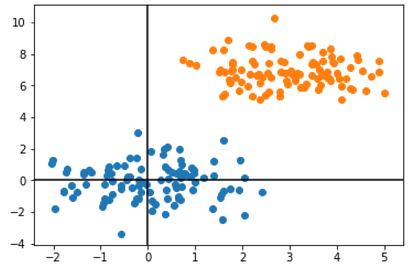
$$\text{prediction} = p_i, \text{ known value} = c_i, \text{ learning rate} = \eta$$
$$\text{error} = \epsilon_i = (c_i - p_i)$$
$$w_i = w_i + \eta * (\epsilon_i * X_i)$$

To actually implement these mathematical ideas in Python code, I created a class which generates Perceptron neural networks fit to analyze data for a given dimension, learning rate, and desired number of epochs, or cycles, to train the network with.

The class contains a prediction function, which predicts the grouping of a datapoint by dotting its inputs with the array of weights. It also contains a weight-training function which performs the above-described backpropagation in order to update the array of weights.

Application

To test the neural network's ability to classify linearly separable data, I generated two groups of data- gaussians with means at (0,0) and (3,7) marked by blue and orange points, respectively.



For visual clarity, I only generated 100 datapoints in each group pictured above- however, in the training set I generated groups with 1000 points each. The learning rate was set to 0.5, and the neural network trained itself over 10 epochs (or cycles).

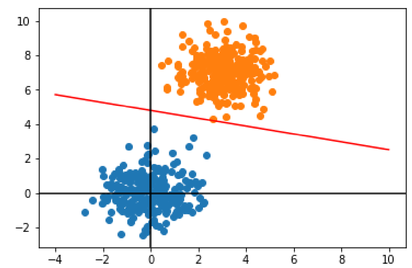
Since the Perceptron classifies linearly separable data, it is instructive to investigate the particular linear boundary the network draws. This can be found by examining the form of the network's output in the 2 dimensional case:

$$w_1 x_1 + w_2 x_2 + w_0$$

Setting this equal to 0 allows us to obtain the function of a line:

$$y = \frac{-w_1}{w_2} x - \frac{w_0}{w_2}$$

By plotting this line, we see the decision boundary generated by the neural network – any point below the line will be classified as belonging to the blue group, whereas any point above will be classified as part of the orange group.



In this case, the neural network predicted the group of each datapoint with 100% accuracy- a promising start! But this doesn't exactly look impressive- a human looking at this data could easily draw that line. To further test the neural network, we should apply it to real data that a human could not obviously visualize.

Application and Conclusion

From here, what might normally be the "meat" of the project is essentially trivial- the application to the neural network to a real, linearly separable dataset works just as simply as the application to the generated dataset above. I decided to apply the network to a dataset of breast cancer data - specifically, data on hundreds of cells with 30 characteristics each, and with each cell classified as cancerous or benign. Although the dataset is not directly tied to the physics questions we've been exploring in this course, it still represents a monumentally important task which can be solved by the type of neural network I'm employing in this project. The dataset was found on the Center for Machine Learning and Intelligent Systems website, and is linked in the Acknowledgements section.

Unfortunately, there is no real way to present this dataset with an attractive plot- and that's a big reason why neural networks are so impressive and useful. Humans clearly cannot visualize a 30-dimensional scatterplot separated by a 29-dimensional hyperplane, but this neural network modeled after a neuron can be applied to classify this dataset.

After being trained on only 250 datapoints for 100 epochs with learning rate 0.01, taking only a few seconds of computational time, **the neural network was able to classify the rest of the datapoints with 91% accuracy.**

This clearly exhibits the power of these mathematical structures- the ability to quickly classify real, complicated data with only the simple framework that separated the two gaussians examined earlier is astounding. Although the perceptron is modeled after a human neuron, and is probably the simplest neural network conceivable, its powers of prediction have immediately outpaced the multi-dimensional data analysis abilities of a human.

By creating a single-layer Perceptron neural network, I was able to obtain a fundamental understanding of the processes underpinning neural networks as a whole. The datasets I would be interested in analyzing in my future are very likely not linearly separable, and will thus require the use of a more complicated neural network- but by completing this exercise, I feel I have equipped myself with the tools and knowledge to begin using more complicated neural networks to analyze data of interest to me, such as astronomical data.

Acknowledgements

https://www.bogotobogo.com/python/scikit-learn/Perceptron_Model_with_Iris_DataSet.php

https://medium.com/@thomascountz/19-line-line-by-line-python-perceptron-b6f113b161f3

https://medium.com/@thomascountz/calculate-the-decision-boundary-of-a-single-perceptron-visualizing-linear-separability-c4d77099ef38

https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53

https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)/

https://machineslearn.co.uk/posts/artificial-neurons-rosenblatts-perceptron/