Neuro-Clustering

Kathan Kashiparekh

March 28, 2018

1 Introduction

Neuro-clustering is a technique to perform an unsupervised machine learning task called 'Clustering' using Neural Networks. Neural networks usually require supervised data in order to learn their weights but this technique provides a unique way to perform unsupervised tasks using neural networks. The model architecture is quite similar to a Perceptron and has just an input layer and an output layer. Number of inputs are determined by the dimensionality of the data while the number of outputs are determined by the number of clusters that are required. An identity activation function is used to propagate the signals exactly as they are received. Input to the output neuron is determined by the Euclidean distance between the input vector and the weights corresponding to the required output neuron. The ouput class is determined by the neuron with the least Euclidean distance. It tries to emulate the K-Means Clustering technique in some way and differs mainly in how the centroids of the clusters are chosen and updated.

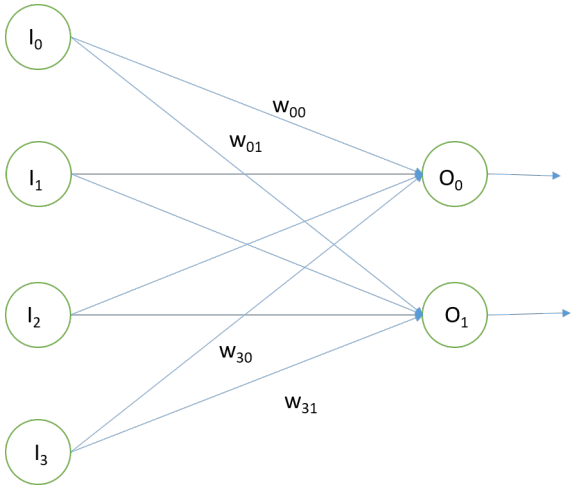

Figure 1: Basic Architecture used for Neuro-Clustering

Figure 1 shows the basic architecture used for Neuro-Clustering. According to the figure, dimensionality of inputs is 4 and data is being divided into two clusters. The weights basically act as cluster centers and the aim is to get the weights as close to the actualy cluster centers as possible. Some useful equations for the same are given below.

Input to output layer: $O_0^{in} = \sum_{i=0}^{3}(w_{i0} - I_i)^2$

Output from the output layer: $O_0^{out} = \sum_{i=0}^{3}(w_{i0} - I_i)^2$

Update equation: $w_{ij} = w_{ij} + \eta * (I_i - w_{ij})$ where j is the index of the output node for which the Euclidean distance is minimized. Here, j is either 0 or 1. And $\eta$is the learning rate.

After the network is trained for a given number of epochs, the clusters are identified via forward propagation of the inputs and cluster label is the index for which the Euclidean distance is minimized.

The weights are initialized using a normal distribution with mean and standard deviation taken from the original dataset,i.e the weights have a mean and standard deviation equal to the one seen in the dataset. This helps the model to converge faster.

2 Experiment 1: Simple dataset

The first experiment corresponds to a self generated dataset using Gaussian distributions with clusters being distinctly visible as seen in Figure 2.

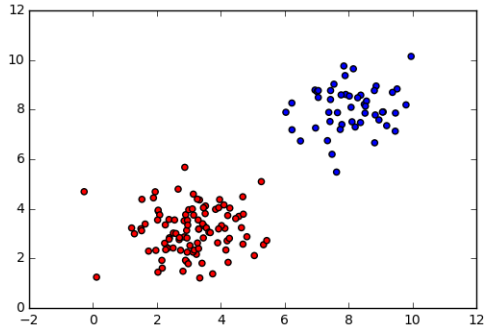

Figure 2: Original Dataset

After running the neuro clustering algorithm for 10 epochs, the following clusters are identified. It can be clearly seen from Figure 3 that the network correctly clusters all the points. Figure 4 shows the how far the weights are from the original cluster centers at each epoch. It can be seen that this stabilizes after

only 2-3 epochs and thus the remaining epochs serve no usefule purpose. Each colour corresponds to one cluster.

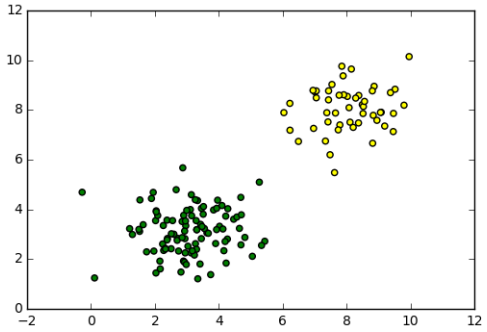

Figure 3: Dataset after being clustered using the algorithm

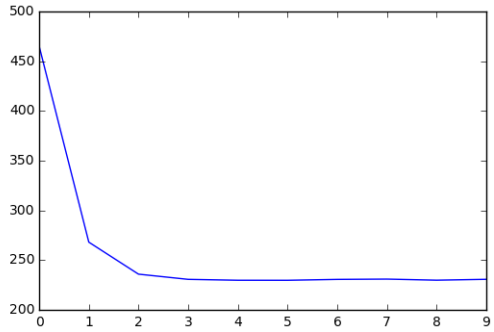

Figure 4: How far the weights are after each epoch?

3 Experiment 2: A bit more complex :P

It is observed from the first experiment that the algorithm classifies the data points perfectly. In order to accurately judge the performance, lets tweak the data a bit. The data now is more cluttered than before with the clusters overlapping a little. Lets see how the network performs on this data.

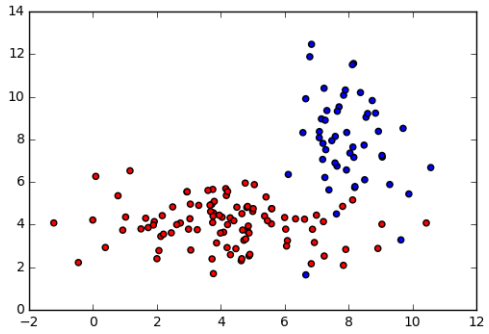

Figure 5: Original complex dataset

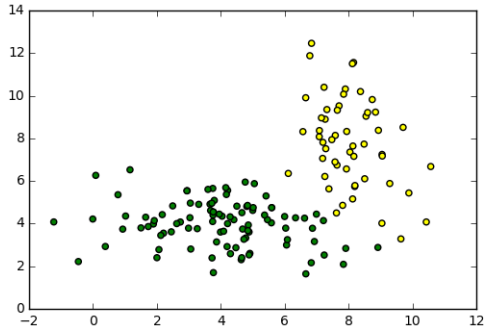

Figure 6: Clustered complex dataset

It can be observed from the clustered dataset that the network is able to cluster most of the points correctly barring a few overlapping points which the given network is bound to mis-classify. The weights decrease in a similar fashion to the first experiment and satuarate after a few epochs.

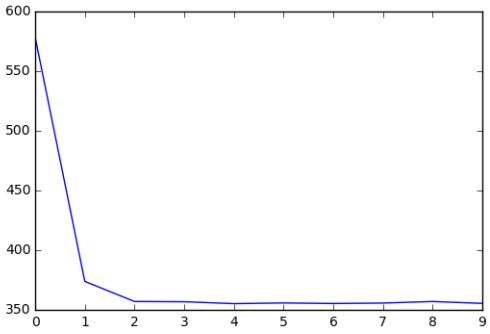

Figure 7: How far are the weights after each epoch?

4 Experiment 3: Iris dataset

The Iris dataset is one of the most famous datasets used to check the validity of very simple and naive classification algorithms. We will try to use neuro-clustering on this dataset to check if the model clusters the flowers analogous to the given class labels. The dataset has 4 features and 3 classes. More details about the dataset can be found here.

The original dataset is plotted in the Figure 8. Dimensions 2 and 3 are chosen to see the clusters a bit more clearly as visualizing all the 4 dimensions is difficult.

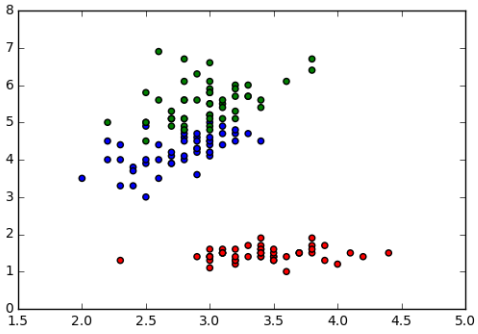

Figure 8: Original Iris dataset using two dimensions

After applying the neuro-clustering algorithm,the results obtained can be seen in Figure 9. On comparing the original dataset and the clustered dataset, it can be seen that barring a couple of points, the network is able to classify

the points correctly. Similar to above experiments, the weights satuarate after a couple of epochs which can be seen in Figure 10,

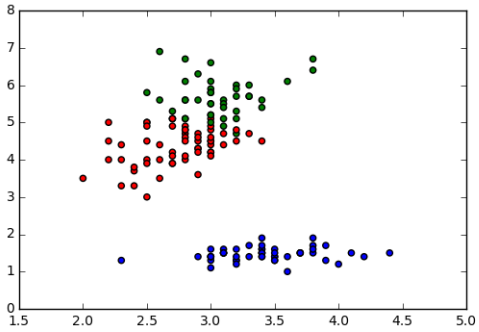

Figure 9: Clustered Iris dataset plotted using same two dimensions

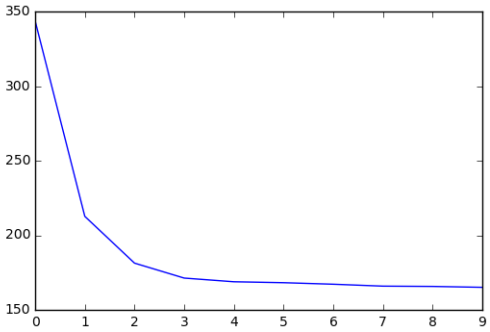

Figure 10: How far are the weights after each epoch?

5 Conclusion

Using a simple perceptron based architecture, we can create a neural network based classifier which performs almost analogously to the K-Means clustering algorithms and reaches the optimum point quickly. Further tests can be run using more complex datasets to check the validity of the algorithm. However, its can be safely said that the given idea works quite accurately for simple datasets. The code for the same can be found on my GitHub profile here.