# Single Neuron Cluster Detection using BCM and Oja Learning Rules

**Michael Campiglia**[*]
*Department of Computer Engineering, Manhattan College*

**Abstract.** Understanding the process in which neurons learn has been a continuously growing area of research throughout recent years. This study seeks to explore two learning models for sensory neurons and apply them to the machine learning task of clustering. More specifically, an algorithm was created to detect clusters in datasets using model BCM and Oja neurons. This algorithm was designed to work well with linearly seperable datasets.

## Introduction

One key topic in the study of machine learning and artificial neural networks is determining how neurons learn [1]. If you take a look at the anatomy of the neuron, as shown in Figure 1, you will find a few key components that make communication and learning possible. The first major component is the axons, which allow the neurons to send outgoing signals. Then there are the dendrites and soma, which receive and store incoming signals, respectively. Finally, there is the synapse, where the axon from one neuron meets the dendrites of another, and communication takes place. As this communication occurs, presynaptic signals sent from the axons impact the postsynaptic action potential in the soma, and if the potential rises high enough, the target soma generates its own action potential and the neuron is said to 'fire'. In the past few decades, extensive research has been conducted on this field of interest, and specific learning rules have been formulated to model the changing connections, or synaptic weights, between neurons.

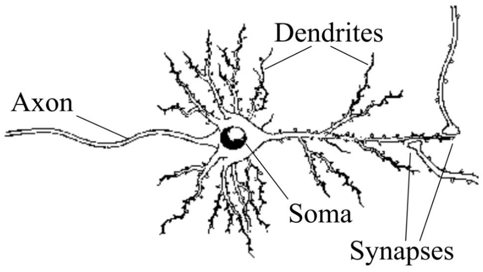

**Figure 1.** Anatomy of a Neuron. (Credit: Bear et al [14])

Knowledge on how neurons learn in biology plays an essential role in machine learning and artificial intelligence, since models developed in neuroscience have fueled these fields. In fact, the basis of the growing field of unsupervised learning is the Hebbian learning theory, which proposes that the strength of connection between neurons is proportional to the degree of correlation

[*]Research mentored by Lawrence Udeigwe, Ph.D.

between pre and post synaptic activity. In other words, if neuron A continuously takes a part in firing neuron B, a change will occur in the cells such that efficiency of A in firing B is increased [2]. Hebbian theory provides the foundational learning algorithm for neural and artificial network models, providing a method to determine how to alter the snyaptic weights between model neurons. Hebb's rule has also resulted in the development of new sensory learning models, such as the two explored in this study, the BCM and Oja learning rules.

These mathematical efforts and learning rules have played a huge part in allowing researchers to continue to model and study both artificial and biological neural networks. One prime example is exhibited through the use of BCM theory in an experiment to model monocular deprivation and the concept of ocular dominance plasticity, the idea that brief deprivation of vision in one eye directly reduces neuronal responses in the closed eye and increases responses in the open one, in mice [4]. In this particular experiment, the BCM learning rule modeled the ocular dominance shift during reverse suture, which is the process of allowing vision in only one eye for a period of time and then reversing the orientation. This process resulted in the portrayal of two key effects, including noise replacing the original retinal activity in the newly closed eye, and a sharp reduction of cortical activity in the neuron, demonstrating the concept of synaptic plasticity. Another example is how the Oja rule's property of being a principal component analyzer (PCA) has led to defining the Oja rule for a layer of parallel neurons and modifiying the Oja rule to perform independent component analysis (ICA), which can find statistically independent components in a dataset[5][6].

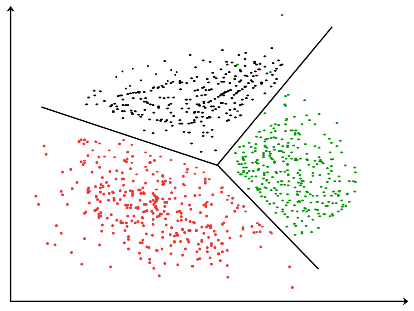

**Figure 2.** Example of Data Clustering. [15]

In the current study, we work on adapting these models to detect clusters in a dataset. As defined in [3], clusters consist of objects or data points that are in some way similar amongst themselves, and clustering is the process of dividing data into these specific groups of similar objects. Clustering is a very active research field involved in statistics, data mining, and machine learning, and implementing efficient clustering algorithms is essential in handling the huge amounts of data that today's technology has allowed us to explore. An example of data point clustering is shown in Figure 2 . This paper will first explore how to model a neuron that can 'learn', and then focus on using this artificial neuron to detect clusters in datasets.

## Methods

**How to model a neuron** One of the simplest ways to model a neuron comes from a generalization of Hebb's Rule and results in a linear neuron, $y = \mathbf{w} \cdot \mathbf{x}$, where

$$\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix} \text{ and } \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \tag{1}$$

In this equation, $\mathbf{x}$ is a vector set of input stimuli, $\mathbf{w}$ is a vector of all the synaptic weights associated to each input, and $y$ is the sum of all the synaptic responses. This version, and actually all neuron models using Hebb's rule are unstable, which is why neural networks usually rely on other stabilized learning models, such as BCM theory and Oja's rule.

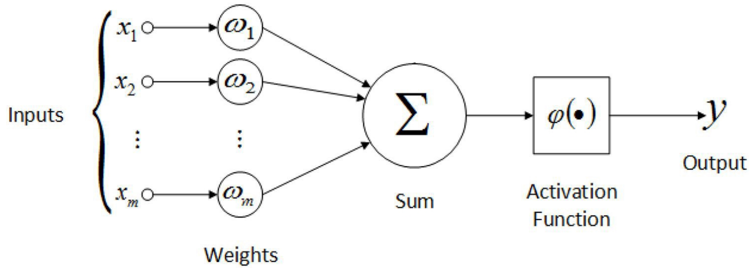

**Figure 3.** Artificial Linear Model Neuron. [16]

**BCM Theory** The BCM theory originates from the 1982 proposal on the theory of neuron selectivity by Bienenstock, Cooper, and Munro [7]. In this formulation, Hebb's rule is stabilized by incorporating a nonlinear function of the postsynaptic activity $\phi(y)$, that changes its sign at a specific value $\theta$, called the modification threshold. If the response $y$ is above $\theta$, the weight vector $\mathbf{w}$ is driven in the direction of the corresponding input $\mathbf{x}$, and driven opposite the direction of the input if the output is below $\theta$. This behavior results in a temporal competition between incoming input patterns [8]. However, the key to BCM theory that solves the instability problems of Hebb's rule is making the modification threshhold $\theta$ a super-linear function of the neuron's output, meaning that it 'slides' with the activity of the neuron. With this addition, the threshhold is able to act as a negative feedback system, allowing $\theta$ to stay ahead of the growth caused by synaptic activity. An example of the super-linear $\theta$ is shown in Figure 4.

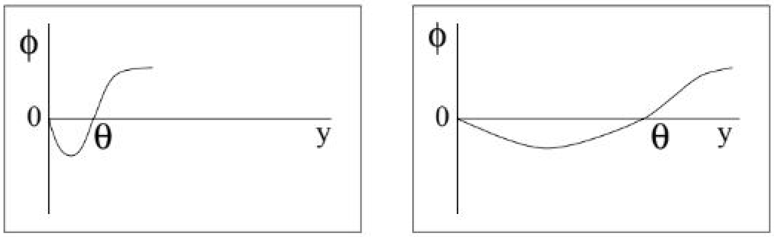

**Figure 4.** Super-linear function $\theta$. [8]

While there are many forms of the $\phi$ function, the one used in this study follows the ordinary differential equation form of the BCM modification equation, resulting in:

$$y = \mathbf{w} \cdot \mathbf{x} \tag{2}$$
$$\dot{\mathbf{w}} = \eta y(y - \theta)\mathbf{x} \tag{3}$$
$$\dot{\theta} = 1/\tau(y^2 - \theta) \tag{4}$$

When looking at the one dimensional model, we get a stable fixed point of $y = \theta = 1$ Evaluating this, $y = 1 = wx$, $w = 1/x$, meaning that the weight compensates depending on the size of the input. In order to simulate the BCM learning rule, the equations can be discretized to

$$y^{(n)} = \mathbf{w}^{(n)} \cdot \mathbf{x}^{(n)} \tag{5}$$
$$\theta^{(n+1)} = \theta^{(n)} + 1/\tau(y^2 - \theta^{(n)}) \tag{6}$$
$$\mathbf{w}^{(n+1)} = \mathbf{w}^{(n)} + \eta y(y - \theta^{(n+1)})\mathbf{x}^{(n)} \tag{7}$$

where $n$ represents the current iteration, $\eta$ is the learning rate and $\tau$ is the threshold averaging constant (sometimes referred to as the memory constant).

**Oja's Rule** Oja's rule was also developed as a variant of the Hebb's rule to avoid the problems that may arise from the instability of Hebb's equation. To fix the instability problem, a decay term is included on the weights, resulting in the form $\dot{\mathbf{w}} = y\mathbf{x} - y^2\mathbf{w}$. With Oja's Rule, increases in connection strength for certain synapses are accompanied by decreases in connection strength for other synapses of the same neuron, which results in spatial competition between inputs [9]. The equations for Oja's rule are described below:

$$y = \mathbf{w} \cdot \mathbf{x} \tag{8}$$
$$\dot{\mathbf{w}} = y\mathbf{x} - y^2\mathbf{w} \tag{9}$$

In order to perform simulations with Oja's rule, the equations are also discretized to

$$y^{(n)} = \mathbf{w}^{(n)} \cdot \mathbf{x}^{(n)} \tag{10}$$

$$\mathbf{w}^{(n+1)} = \mathbf{w}^{(n)} + \eta(yx^T - y^2\mathbf{w}) \tag{11}$$

where $\eta$ is the learning rate.

**Detecting clusters using sensory neurons** To have a better understanding for the idea of detecting clusters using an artificial neuron, it is important to take a look at two related concepts for neurons. One is synaptic plasticity, which states that the strength of connection between two neurons can change over time [12]. The other is neuronal selectivity, which states that over a period of training, a neuron learns to react and fire strongly to certain stimuli while reacting weakly to others [13].

The idea of detecting clusters using the BCM and Oja learning rules on an artificial neuron is based on the premise that if a neuron becomes selective to a stimulus input, then that neuron should also be selective to every other stimulus near it. If we consider a data point to be treated as a stimulus input, we can then translate this concept to a dataset and group nearby data points into clusters. With this idea in mind, we can form the basis of the algorithm by focusing on grouping the responses of the artificial neuron.

For both BCM and Oja's rule, the neuron is trained by running many iterations of the discretized learning rule equations on a dataset of inputs, $\mathbf{X}$. The dataset consists of a $m$ x $n$ matrix, where $m$ represents the number of rows of inputs and $n$ represents the dimensionality of the input. Thus, each row of $\mathbf{X}$ represents a data point $\mathbf{x}$. For each iteration, a row of $\mathbf{X}$ is randomly chosen from the full set of inputs, and the response becomes $y = \mathbf{w} \cdot \mathbf{x}$. Over time, the synaptic weight vector $\mathbf{w}$ will adjust itself from the learning rule and will ultimately become selective to an input, or in this case, a cluster of inputs.

## Algorithm

The cluster detection algorithm was written and tested in MATLAB. The function takes in two parameters, the matrix of inputs $\mathbf{X}$, described earlier, and the number of clusters expected to be in the data set, denoted *Clusters* . The algorithm then begins a loop, running for a set number of iterations, *numItr*. Each iteration, a row of $\mathbf{X}$, which represents a stimulus input, is chosen at random to train the neuron using either the BCM or OJA learning rule, depending on the user's choice. As each iteration takes place, the resulting synaptic response and weight vectors are stored into the matrices $Y$ and $W$ respectively, where $Y$ has the same number of rows as inputs, $W$ has $m$ rows and $numItr$ columns, and $W$ has $n$ rows and $numItr$ columns, referring back to the parameters mentioned earlier while describing the $m$ x $n$ matrix.

Next, the algorithm takes the last column of *Y*, which represents the last iteration of the synaptic responses for all of the inputs, and performs a sort on this data, storing it in a new array, *YSort*.

After the responses are sorted, a for loop iterates through each index of the array and the difference between each value and its predecessor is recorded and stored in the array *YDiff*. The maximum value in *YDiff* is recorded as *maxDiff*, which represents the largest jump between two synaptic responses and is thus a demarcation between clusters.

After this, the index of *maxDiff* within the *YDiff* array is found. The value of *YSort* at this index is stored in the variable *BreakPoint*, as this represents the value of the last synaptic response before a cluster changes. After this is complete, a new for loop iterates throughout the last column of $Y$, this time splitting up the data depending on whether the index is greater or less than the *BreakPoint*. The smaller group is considered to be the first cluster of the data, and the inputs in the cluster are removed from $\mathbf{X}$. After this, the remaining inputs are kept within the data set, and the algorithm is repeated until it reaches the predefined number of clusters.

**Algorithm 1** Train BCM Neuron with dataset $\mathbf{X}$

```
Initialize $\theta = 0$, $w =$ a non-zero vector
for $k = 1$ to $numItr$ do
  $\mathbf{x} \leftarrow$ random row of $\mathbf{X}$
  $y \leftarrow \mathbf{x} \cdot \mathbf{w}$
  $\mathbf{w} \leftarrow \mathbf{w} + \eta y(y - \theta)\mathbf{x}^T$
  $\theta \leftarrow \theta + \frac{1}{\tau}(y^2 - \theta)$
end for
$Y \leftarrow \mathbf{X} \cdot \mathbf{w}$
```

**Algorithm 2** Train Oja Neuron with dataset $\mathbf{X}$

```
Initialize $w =$ a non-zero vector
for $k = 1$ to $numItr$ do
  $\mathbf{x} \leftarrow$ random row of $\mathbf{X}$
  $y \leftarrow \mathbf{x} \cdot \mathbf{w}$
  $\mathbf{w} \leftarrow \mathbf{w} + \eta(y\mathbf{x}^T - y^2\mathbf{w})$
end for
$Y \leftarrow \mathbf{X} \cdot w$
```

## Results

For all of my results, I ran the datasets through 500,000 iterations, with parameters $\eta = 0.01$ and $\tau = 10$. I began testing the cluster detection algorithm by using an artificially created 2D data set as my set of inputs. I created three clusters, each consisting of 40-70 data points, distinctly seperated from one another at different locations of the x-y plane. With the BCM learning rule, the clustering algorithm worked extremely well with this test set, detecting the clusters to be what one expect them to be by looking at them. This is shown in Figure 5.

**Algorithm 3** Cluster Detection on data set X

```
for $k = 1$ to Clusters -1 do
  Train neuron using algorithm 1 or 2 to return Y
  YSort $\leftarrow$ sort(Y)
  for $i = 1$ to Size(Y) -1 do
    YDiff $\leftarrow$ Take difference of consecutive Y indeces
  end for
  maxDiff $\leftarrow$ max(YDiff)
  maxIndex $\leftarrow$ Find index where YDiff = maxDiff
  BreakPoint $\leftarrow$ value of YSort at maxIndex
  for $i = 1$ to Size(Y) do
    if Y(i) $\leq$ BreakPoint then
      Store into LowerGroup
    else
      Store into UpperGroup
    end if
    if LowerGroup $>$ UpperGroup then
      Remove UpperGroup inputs from X and store into new Cluster
    else
      Remove LowerGroup inputs from X and store into new Cluster
    end if
  end for
end for
```

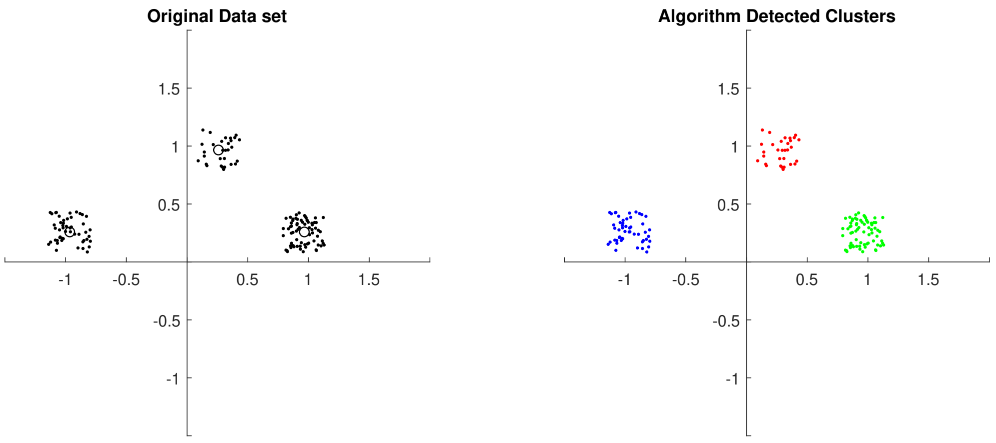

**Figure 5.** Original 2D data set vs. Algorithm defined clusters using BCM.

I also tested a similar dataset with an Oja neuron, with results exhibited in Figure 6.

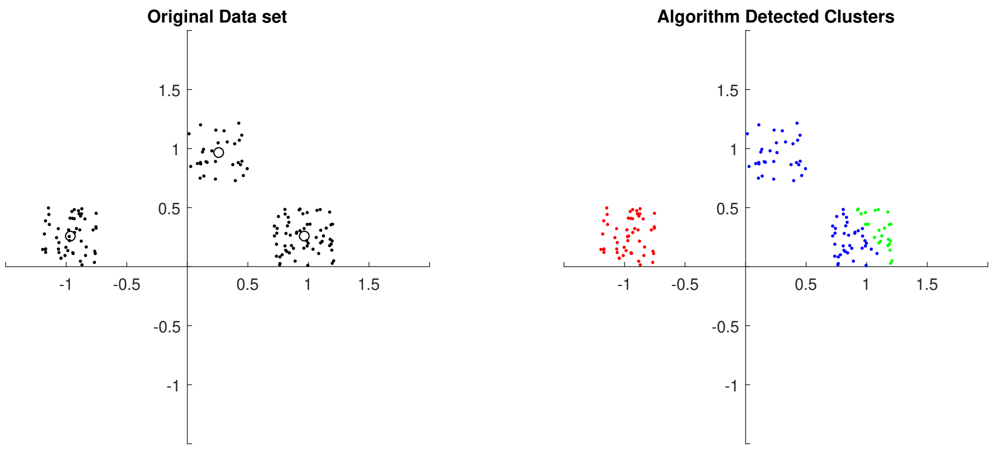

**Figure 6.** Original 2D data set vs. Algorithm defined clusters using OJA.

Next, an artificial 3D data set was created and I performed the cluster detection algorithm using BCM and Oja's rule, as shown in Figure 7 and Figure 8.

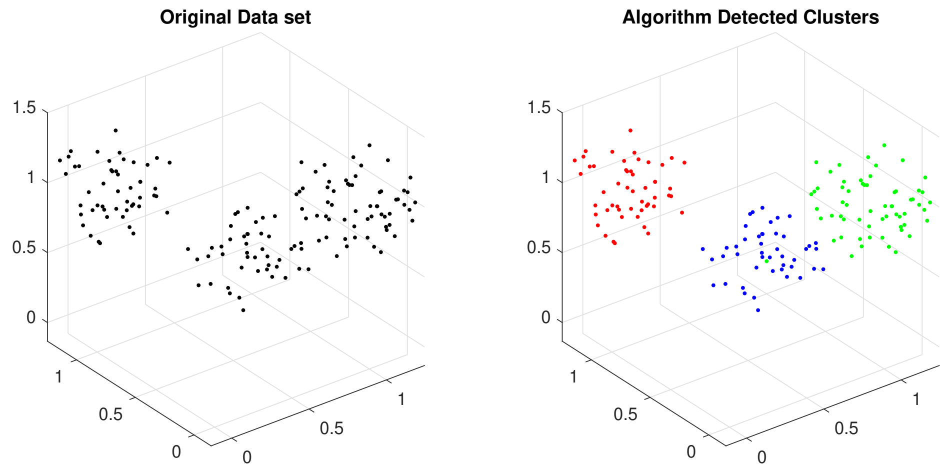

**Figure 7.** Original 3D data set vs. Algorithm defined clusters using BCM.

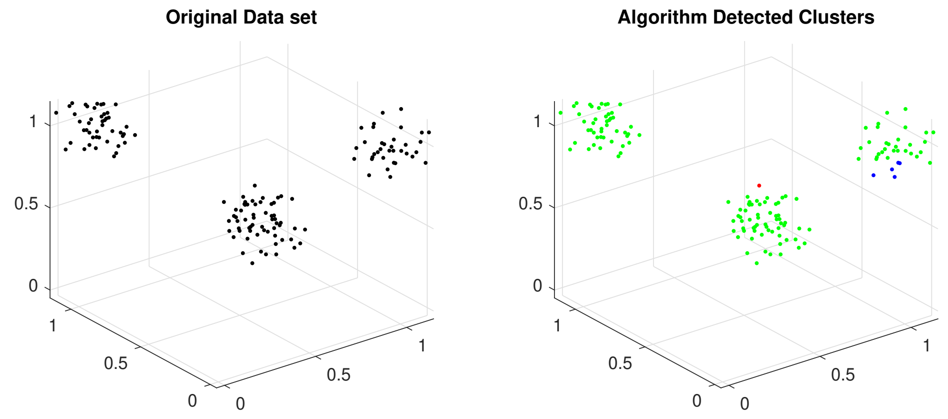

**Figure 8.** Original 3D data set vs. Algorithm defined clusters using OJA.

Finally, I tested the algorithm on a real dataset from the UCI Machine Learning Repository [17]. The first real dataset that I tested was the Iris dataset. This dataset included five variables, with the first four representing the lengths and widths of the petals and sepals of Iris flowers, and the fifth variable representing their class type, in this case being either Iris Setosa, Iris Versicolor, or Iris Virginica. The dataset ran through the algorithm with all four variables, but since you cannot visualize a four dimensional dataset, only the first three variables are plotted in Figure 9 and Figure 10 below.

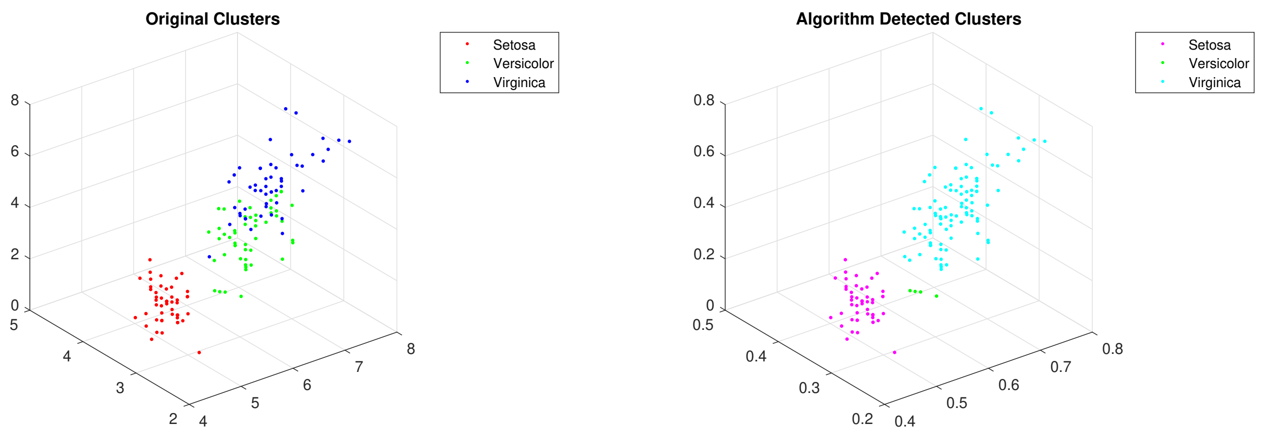

**Figure 9.** Original Iris Data set classes vs. Algorithm defined clusters using BCM.

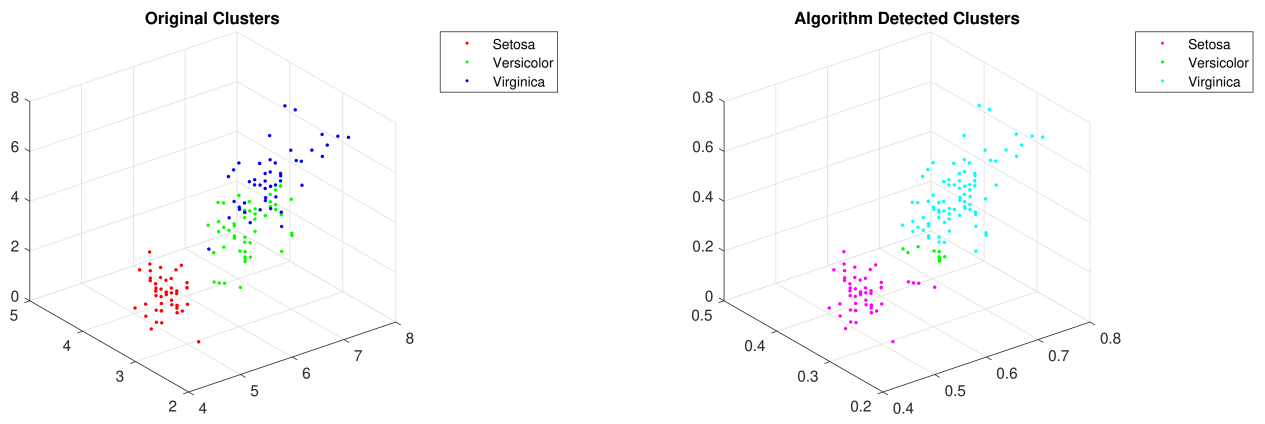

**Figure 10.** Original Iris Data set classes vs. Algorithm defined clusters using OJA.

The other real dataset that I tested was the Seeds dataset. This dataset included eight variables, with the first seven representing the various parameters of the seeds, including the area, perimeter, compactness, asymmetry, etc., and the eighth variable representing their class type, in this case being either type 1, 2, or 3. The dataset ran through the algorithm with all seven variables, but to visualize the clusters, only three of the variables are plotted in Figure 11 and Figure 12 below.

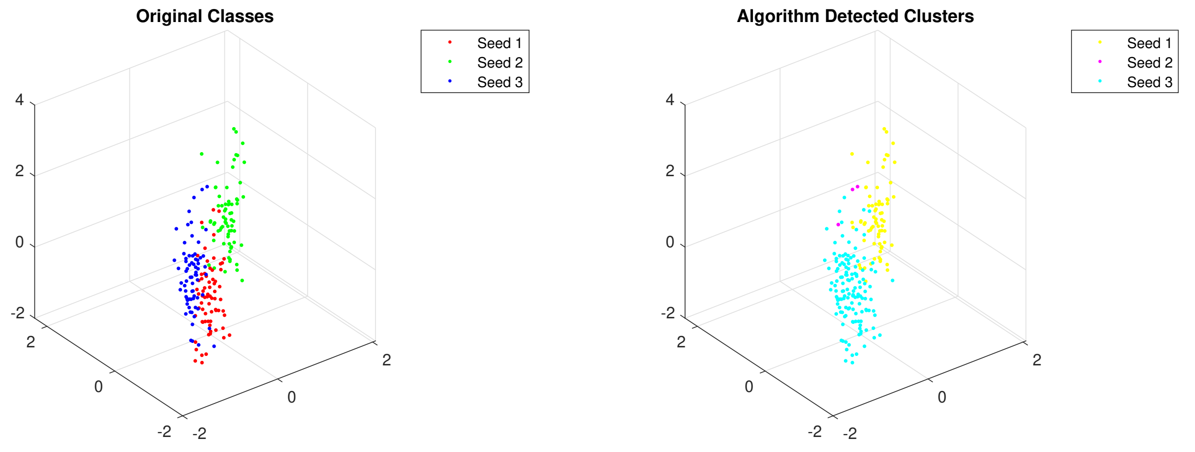

**Figure 11.** Original Seeds Data set classes vs. Algorithm defined clusters using BCM.

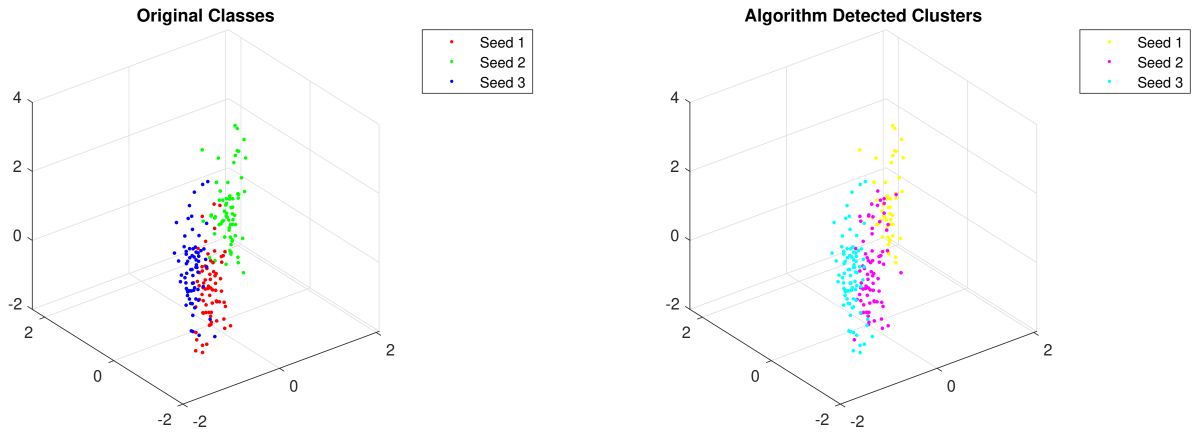

**Figure 12.** Original Seeds Data set classes vs. Algorithm defined clusters using OJA.

## Discussion

For the most part, the results of this study supported the idea that data points near each other would trigger similar levels of synaptic responses, and therefore would be grouped into clusters after being ran through the algorithm. The clearest example of this is displayed in Figure 5, where the distinctly separated groups of data were grouped into visually expected clusters using the BCM neuron. One reason for these results on this artificial dataset is because the clusters are linearly separable, meaning that a plane exists which can be used to separate each cluster from the rest of the dataset [9]. It was also found that the best results for all datasets occurred when $\eta = 0.01$ and $\tau = 10$.

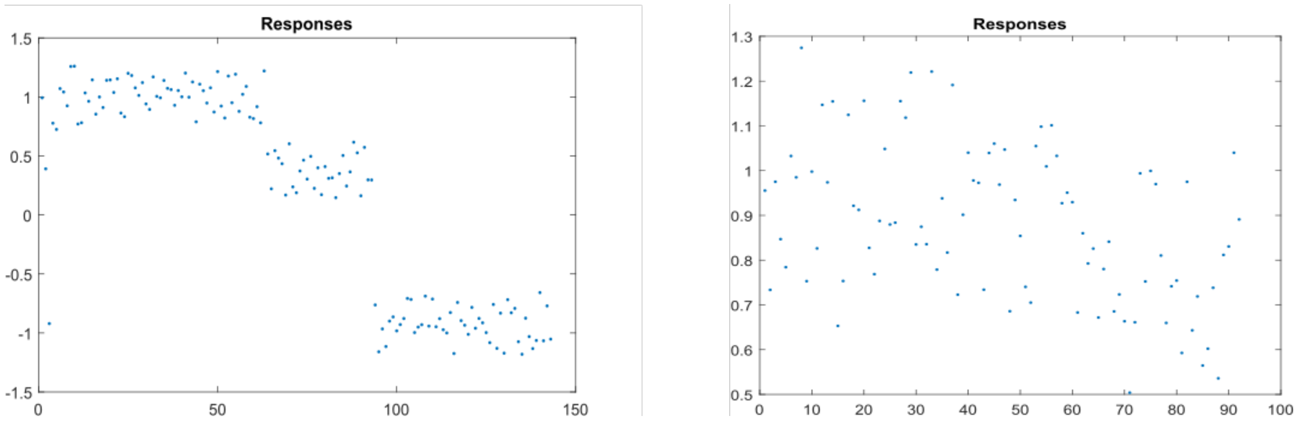

**Figure 13.** Original synaptic responses vs. Responses after a cluster was removed

In both artificial datasets for the Oja neuron, Figure 6 and Figure 8, the clusters formed by the algorithm did not match with how one would group the data visually. To look further into why this was happening, I plotted the synaptic responses of all the inputs, organized in a way so that the visual clusters followed one another, during each iteration of the algorithm. After completing this, it was found that the synaptic responses of the groups were originally similar to one another and distinctly unique from the other clusters, supporting the idea of neuronal selectivity described earlier. However, it was discovered that after the first cluster was removed from the dataset and the neuron was retrained, the remaining synaptic responses did not show any selectivity, which can explain why the algorithm did not produce expected results for the Oja neuron. These findings are displayed in Figure 13 above. It was also found that in both neurons for the Iris dataset, the cluster that was distinctly seperate from the rest of the data was detected with the algorithm, while the rest of the data formed slightly different clusters each test.

The work completed in this study can serve as a basic feature detection tool and can help compress larger datasets by grouping certain columns into clusters. There are many situations where having an extra column which groups multiple attributes in a dataset can be very useful, such as in analyzing patterns or presenting statistical results to clients in a way that is easier to visualize and understand. One example of this applciation is seen in [10][11], where Intrator et al. combined well-known statistical feature extraction models with the BCM learning rule to develop an algorithm for recognizing 3D objects from a 2D input image. One path of future work that could be taken with this study is exploring the implications and details of implementing a lateral inhibition network of BCM neurons and observing the similarities and differences with the new system. Another potential area of improvement would involve modifying the algorithm to handle other data types other than purely numerical data, as well being able to handle data that does not have the same number of attributes across the entire data set. We also plan to find ways to improve the Oja neuron results, with tactics such as weight or input normalization.

## Acknowledgments

This research was supported by the School of Science Research Scholars Program. The author would like to thank Dr. Lawrence Udeigwe for his support and guidance throughout the research.

## References

[1] Oja, Erkki."Oja Learning Rule." Scholarpedia, vol. 3, no. 3, 2008, p. 3612, 10.4249/scholarpedia.3612.

[2] Klein, Raymond. "Donald Olding Hebb." Scholarpedia, vol. 6, no. 4, 2011, p. 3719, 10.4249/scholarpedia.3719.

[3] Berkhin, Pavel. "A survey of clustering data mining techniques." Grouping multidimensional data. Springer, Berlin, Heidelberg, 2006. 25-71.

[4] Frenkel, Mikhail Y., and Mark F. Bear. "How monocular deprivation shifts ocular dominance in visual cortex of young mice." Neuron 44.6 (2004): 917-923.

[5] Oja, Erkki. "Principal components, minor components, and linear neural networks." Neural networks 5.6 (1992): 927-935.

[6] Oja, Erkki. "PCA, ICA, and nonlinear Hebbian learning." Proc. Int. Conf. on Artificial Neural Networks (ICANN'95). 1995.

[7] Bienenstock, Elie L., Leon N. Cooper, and Paul W. Munro. "Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex." Journal of Neuroscience 2.1 (1982): 32-48.

[8] Blais, Brian S., et al. The role of the environment in synaptic plasticity: towards an understanding of learning and memory. Diss. Brown University, 1998.

[9] Udeigwe, Lawrence C., Paul W. Munro, and G. Bard Ermentrout. "Emergent dynamical properties of the BCM learning rule." The Journal of Mathematical Neuroscience 7.1 (2017)

[10] Intrator, Nathan, et al. Three Dimensional Object Recognition Using an Unsupervised Neural Network: Understanding the Distinguishing Features. No. TR-63. Brown Univ Providence RI Inst for Brain and Neural Systems, 1992.

[11] Intrator, Nathan. "Feature extraction using an unsupervised neural network." Connectionist Models. Morgan Kaufmann, 1991. 310-318.

[12] Abbott, Larry F., and Sacha B. Nelson. "Synaptic plasticity: taming the beast." Nature neuroscience 3.11s (2000): 1178.

[13] Frégnac, Yves, and Michel Imbert. "Development of neuronal selectivity in primary visual cortex of cat." Physiological Reviews 64.1 (1984): 325-434.

[14] Bear, Mark F., Barry W. Connors, and Michael A. Paradiso, eds. Neuroscience. Vol. 2. Lippincott Williams and Wilkins, 2007.

[15] R. Ng and J. Han. "Efficient and effective clustering method for spatial data mining". In: Proceedings of the 20th VLDB Conference, pages 144–155, Santiago, Chile, 1994.

[16] Jayesh Bapu Ahire. Ostojic, Srdjan, and Nicolas Brunel. "From spiking neuron models to linear-nonlinear models." PLoS computational biology 7.1 (2011): e1001056.

[17] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California.