An Analysis of the Cluster-Detecting Property of the BCM Neuron

Lawrence C. Udeigwe

Department of Mathematics, Manhattan College
4513 Manhattan College Parkway
Riverdale, NY 10471, USA
lawrence.udeigwe@manhattan.edu
http://manhattan.edu/campus-directory/lawrence.udeigwe

The BCM learning rule was first proposed by Elie Bienenstock, Leon Cooper, and Paul Munro to measure the selectivity of neurons in the primary visual cortex and its dependency on neuronal inputs. We show that an artificial BCM neuron has the ability to detect clusters in a dataset. While paying attention to the qualitative behavior of an underlying system of differential equations, we present an analysis of this cluster-detecting property of such a neuron. We also analyze, and discuss the performance of, a resulting clustering algorithm.

Keywords: Neuron, BCM learning rule, Clustering

1. Introduction

It is well corroborated that when trained in the right environment, a sensory neuron become selective; that is, it responds to a certain class of stimuli but not others. Experiments by Hubel and Wiesel, for instance, revealed that neurons of the visual cortex of cats may respond preferentially to stimulus orientation. Their studies also show that these neurons detect edges regardless of where – in the receptive field of the neuron – they are placed, and can preferentially detect motion in certain directions. While all sensory neurons are selective, their preference and degree of selectivity vary.

Synaptic plasticity is an umbrella term used to describe how the strength of the synapse between a neuron and a stimulus – or between a neuron and another neuron – modifies over time. One of the most cited theories in this field is the Hebbian theory of synaptic plasticity. Donald Hebb in 1949 proposed that when a neuron A repeatedly participates in firing neuron B, the strength of the influence of A onto B increases. If we think of A as the pre-synaptic neuron and B as the post-synaptic neuron, the Hebbian theory implies that changes in synaptic strengths in a neural network is a function of the pre- and post-synaptic neural activities. A few decades later, Nass and Cooper developed a Hebbian synaptic modification theory for the synapses of the visual cortex, which was later extended to a threshold dependent setup by Cooper at al. In this setup, the sign of a synaptic strength modification is based on whether the post-synaptic response is below or
above a static threshold. A response above the threshold is meant to strengthen the active synapse, and a response below the threshold should lead to a weakening of the active synapse.

One of the most accurate models of synaptic plasticity to date is the BCM learning rule with which Bienenstock et al (by incorporating a dynamic threshold that is a function of the average post-synaptic activity over time) captured the development of stimulus selectivity in the primary visual cortex of higher vertebrates. A neuron that adheres to the BCM learning rule is called a BCM neuron and a network of a collection of such neurons is called a BCM network. In corroborating the BCM theory, Shouval et al and Intrator and Cooper showed that a BCM network develops orientation selectivity and ocular dominance in a natural scene environment. Although the BCM rule was developed to model plasticity and selectivity of visual cortical neurons, it has been successfully applied to other classes of neurons. For instance, it has been used to explain experience-dependent plasticity in the mature somatosensory cortex. Furthermore the BCM rule has been reformulated and adapted to suit various interaction environments of biological neural networks, including laterally interacting neurons and stimuli generalizing neurons.

Mathematically speaking, the BCM learning rule is a system of differential equations involving the synaptic strengths (also known as synaptic weights), the stimulus coming into the neuron, the activity response of the neuron to the stimulus, and the threshold for the activity. Unlike its predecessors, which use static thresholds to modulate neuronal activity response, the BCM learning rule allows the threshold to be dynamic. This dynamic threshold provides stability to the learning rule, and from a biological perspective, provides homeostasis to the system.

The BCM learning rule have been used to demonstrate a few machine learning applications of neuronal selectivity. Intrator et al showed that a BCM neural network can perform projection pursuit, i.e. it can find projections in which a data set departs from statistical normality. This is an important finding that highlights the feature detecting property of a BCM neural model. As a result, the BCM neural network has been successfully applied to some specific pattern recognition tasks. For example Bachman et al incorporated the BCM learning rule in their algorithm for classifying radar data. Intrator et al developed an algorithm for recognizing 3D objects from 2D view by combining existing statistical feature extraction models with the BCM model. In Ref. 33, Poljovka and Benuskova presents a preliminary simulation on how the BCM learning rule has the potential to identify alpha numeric letters.

In the present work we show that a BCM neuron has the ability to detect clusters in a dataset. In section 2 we discuss the BCM learning rule as a system of stochastic differential equations and explain its variables and parameters as well as how synaptic weigh training is achieved. In section 3 we illustrate a simulation in the simplest case of a single neuron with two weights and two different competing stimuli. We then extend this simulation to competing data point clusters and note
the variations in selectivity behavior that are a function of the sizes of the clusters and the angular distance between the clusters. In section 4 we derive the mean field system of equations for the BCM neuron and show that there are changes in the stability as the synaptic threshold time scale changes. In section 5 we analyze a resulting clustering algorithm; we also analyze its computational complexity, and how it performs in comparison to benchmark algorithms that perform similar tasks.

2. Mathematical methods

We model a neuron with the following variables:

- \( x = \{x_1, x_2, \ldots, x_n\} \), a vector representing the stimulus pattern being received by the neuron.
- \( w = \{w_1, w_2, \ldots, w_n\} \), a vector of synaptic weights by which the components of \( x \) are scaled.
- \( v = w \cdot x \), the response of the neuron to the stimulus \( x \).

This model is usually referred to as the linear model, in that the response is a weighted linear combination of each component of the stimulus pattern. See fig. 1A for a visual representation of the linear neuron model.

For a single neuron, the BCM learning rule expresses the changes in synaptic weights, \( w \) as a product of the input stimulus vector, \( x \), and a nonlinear function, \( \phi \), of the neuronal response, \( v \), and a dynamic threshold, \( \theta \), of the response. If at any time, the neuron receives a stimulus \( x \) from a stimulus set, say \( \{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\} \),
the synaptic weight vector, $w$, evolve according to the BCM rule as follows:

$$\dot{w} = \phi(v; \theta)x$$

$$\dot{\theta} = E_p[v]$$

(1)

where $\dot{w}$ means $\frac{dw}{dt}$. The quantity $\theta$ is sometimes referred to as the “sliding threshold” because, as can be seen from eq. (1), it changes with time, and this change depends on the neuronal response $v$. The function $\phi$ has the following property: for low values of the post-synaptic activity (i.e $v < \theta$), $\phi$ is negative; for high values of the post-synaptic activity (i.e $v > \theta$), $\phi$ is positive. The results presented in Ref. 4 use $\phi(v) = v(v - \theta)$. Fig. 1B shows the behavior of this particular $\phi$. The quantity $E[v]$ is a running temporal average of $v$, and the learning rule is stable for $p > 1$.

Later formulations of the learning rule – for instance the one presented in 24 – have shown that a spatial average can be used in lieu of a temporal average, and that with $p = 2$, $E[v^p]$ is a good approximation of $E_p[v]$. We can also replace the moving temporal average of $v$ with first order low-pass filter. Thus a differential form of the learning rule (as presented in Ref. 12) is

$$\tau_w \dot{w} = vx(v - \theta)$$

$$\tau_\theta \dot{\theta} = (v^2 - \theta)$$

(2)

where $\tau_w$ and $\tau_\theta$ are time scale factors, which in simulated environments, can be used to adjust how fast the system is changing with respect to time. We point out that the vector input, $x$, is changing rapidly compared to $\theta$ and $w$; thus equation (2) is actually a stochastic equation. The stimulus, $x$, are generally taken from a finite set of patterns, $x^{(k)}$ and is randomly selected and presented to the model.

Once its dynamics are at a stable steady state, the BCM neuron is selective. This means that it exhibits high-response activities to a relatively small subset of the stimuli presented to it and exhibits very low-response activities to other stimuli. With data points as stimuli, our conjecture is that points that are close enough – in the Euclidean sense – to each other would yield comparable responses. Thus, based on this conjecture, the main focus of the present work is explore the ability of the BCM neuron to detect a cluster within a set of data points.

3. Preliminary experiments

We proceed to numerically simulate different scenarios of the BCM learning rule that are vital to the present discussion.

3.1. Two stimuli, two weights

Consider a BCM neuron that receives a stimulus input $x$ stochastically from a set $\{x^{(1)}, x^{(2)}\}$ with equal probabilities; that is, $Pr[x(t) = x^{(1)}] = Pr[x(t) = x^{(2)}] = \frac{1}{2}$. We create a simple hybrid stochastic system where the value of $x$ switches between the pair $\{x^{(1)}, x^{(2)}\}$ as in a two state Markov process. Fig. 2 shows results for when $\tau_w = 100$, $\tau_\theta = 10$, the stimuli are $x^{(1)} = (\cos \pi/24, \sin \pi/24)$ and $x^{(2)} = \ldots$
Cluster-Detecting Property of the BCM Neuron

Fig. 2. Left: Two stimuli presented to a BCM neuron. $x^{(1)} = (\cos \pi/24, \sin \pi/24)$ is red and $x^{(2)} = (\sin \pi/24, \cos \pi/24)$ is blue. Right: response of the neuron over time to each of the two stimuli. The colors of the plots correspond to the colors of the stimuli. The initial conditions of $w_1$, $w_2$ and $\theta$ lie in the interval $(0, 0.5)$. $\tau_w = 100$, and $\tau_\theta = 10$.

$(\sin \pi/24, \cos \pi/24)$ – respectively the red dot and the blue dot on the left panel – and the initial conditions of $w_1$, $w_2$ and $\theta$ lie in the interval $(0, 0.5)$. We concede that the choice of $\pi/24$ is arbitrary, however the decision to employ unit vectors for the stimuli is motivated by analyses that will be shown in future sections. The two plots in the right panel of fig. 2 show the response of the neuron to each of the two stimuli over time. The colors of the plots correspond to the colors of the stimuli. As can be inferred from the steady state behavior of the neuron, it becomes selective and chooses $x^{(1)}$ with response of approximately 2 over $x^{(2)}$ which yields a response of approximately 0. We note that since the presentation of stimuli is stochastic, it is possible that if this experiment is repeated the neuron will choose $x^{(2)}$ over $x^{(1)}$.

3.2. Cluster analysis

![Cluster centers defined as pattern vectors](image)

Fig. 3. Cluster centers defined as pattern vectors $x = (\cos \alpha, \sin \alpha)$ and $x = (\cos \alpha, \sin \alpha)$ for (A) $\alpha > 0$ (B) $\alpha > 0$ (C) $\alpha < 0$.

We define a *stimulus cluster* as a group of stimuli with some sort of similarity.
If the stimuli are represented as vectors – as we have been doing – the center of a cluster if thus the mean of the all the vectors in comprising the cluster. Motivated by the setup and results of section 3.1, we construct an artificial set of stimuli whose members have the form $x = (\cos \alpha + \text{noise}, \sin \alpha + \text{noise})$ or $x = (\cos \alpha + \text{noise}, \sin \alpha + \text{noise})$. This is equivalent to creating two clusters with centers $(\cos \alpha, \sin \alpha)$ and $(\cos \alpha, \sin \alpha)$.

According to a theorem in Ref. 4, a BCM neuron is most selective when the set of stimulus patterns is linearly independent. When $\alpha = \pi/4$, the two centers described above are the same. When $\alpha = -\pi/4$, the angle between the two centers is $\pi$, implying that they are opposite of each other, and thus linearly dependent. Therefore it is important to restrict $\alpha$ to $-\pi/4 < \alpha < \pi/4$.

**Lemma 1.** For any two angles $\beta_1$ and $\beta_2$, there exists an angle $\alpha$ such that a uniform rotation converts the set of unit vectors $\{x^{(1)} = (\cos \beta_1, \sin \beta_1), x^{(2)} = (\cos \beta_2, \sin \beta_2)\}$ to $\{x^{(1)} = (\cos \alpha, \sin \alpha), x^{(2)} = (\sin \alpha, \cos \alpha)\}$

**Proof.** It suffices to find the required rotation angle. If this angle is $h$, then $\alpha$ is such rotating $x^{(1)}$ by $h$ gives

$$\cos \alpha = \cos(\beta_1 + h), \quad \sin \alpha = \sin(\beta_1 + h)$$

and rotating $x^{(2)}$ by $h$ gives

$$\sin \alpha = \cos(\beta_2 + h), \quad \cos \alpha = \sin(\beta_2 + h)$$

Now, recall that for any $y$, $\sin y = \cos(\pi/2 - y)$. So setting $\cos \alpha$ of eq. (3) equal to the $\cos \alpha$ of eq. (4) gives $\beta_1 + h = \pi/2 - (\beta_2 + h)$. Thus the required rotation angle is $h = \frac{\pi}{4} - \frac{(\beta_1 + \beta_2)}{2}$ and the required parametric angle for the two vectors is $\alpha = \frac{(\beta_1 - \beta_2)}{2} + \frac{\pi}{4}$. \hfill \square

Thus for the sake of our analysis in this section and beyond, we measure the distance between two stimuli by measuring the non-reflex angle between the two vectors representing the points. The prescribed range of $\alpha$ above captures the desired angular measure because any value outside this range would result in a reflex angle. fig. 3 pictorially explains the relationship between the two centers for $\alpha > 0$ and $\alpha < 0$.

Now let $C_1$ be the cluster whose center is $(\cos \alpha, \sin \alpha)$ and $C_2$ the cluster whose center is $(\cos \alpha, \sin \alpha)$. We parameterized these clusters as follows

$$C_1 = \{x^{(i)} = (\cos \alpha + b \epsilon_i, \sin \alpha + b \epsilon_i) \mid b > 0, -1 < \epsilon_i < 1, i = 1, 2, \ldots, N_1\}$$

$$C_2 = \{x^{(j)} = (\cos \alpha + b \epsilon_j, \sin \alpha + b \epsilon_j) \mid b > 0, -1 < \epsilon_j < 1, i = 1, 2, \ldots, N_2\}$$

where $\epsilon_i$ and $\epsilon_j$ are uniformly distributed and determine the deviation of each stimulus from the cluster center; $b$ is used to control how scattered the clusters are; and $N_1$ and $N_2$ are the respective number of stimuli in the clusters.
We proceed to the train a BCM neuron by presenting to the neuron a stimulus from $C_1 \cup C_2$ at each iteration. Each row of fig. 4 shows the result of carrying out and instance of this simulation. In each instance, the initial conditions of $w_1$, $w_2$ and $\theta$ lie in the interval $(0,0.5)$, $\tau_w = 100$, $\tau_\theta = 10$, and $k = 0.35$. The left column shows the distribution of the stimuli on the $xy$ plane; the middle column shows plots of the responses of the neuron to the cluster centers. As in fig. 2, the colors of the plots correspond to the colors of the cluster centers. The right column shows box plot distributions of the responses.

In fig. 4A, $N_1 = 50$, and $N_2 = 50$. With $\alpha = 0$, the clusters in this case are clearly far apart enough from each other. At steady state, it can be seen that the responses to the centers are stable, and the neuron selects the center of cluster $C_1$. The box plot shows that the responses to each cluster in fact forms its own cluster and that there is no intersection between the clusters of responses. The responses
to the stimulus cluster whose center is selected by the neuron are higher than the responses to the stimulus cluster whose center is not selected.

In fig. 4B, \(N_1 = 50\), and \(N_2 = 50\). With \(\alpha = \pi/12\), the clusters are still far enough from each other, although not as far apart as in 4A. At steady state, it can be seen that the responses to the centers are stable, and the neuron selects the center of cluster \(C_2\). The box plots shows that the responses to the stimulus clusters. A conclusion similar to that made for the box plots in fig. 4A can also be made here.

In fig. 4C, \(N_1 = 70\), and \(N_2 = 30\). With \(\alpha = \pi/12\), the clusters in this case are still far enough from each other. At steady state, it can be seen that the responses to the centers are still stable, and the neuron selects the center of cluster \(C_2\). Note that the response to the center of \(C_2\) here is not as high as the response to \(C_2\) in fig. 4B.

We have shown that as long as the clusters are far enough from each other, the BCM neuron will exhibit a clear difference between the two sets of responses to the two different clusters. Note that the ratio of \(N_1\) to \(N_2\) is a factor in determining the value of the response to the selected cluster center. This will be considered in future analysis.

### 3.3. Oscillatory responses

![Fig. 5. Left: Two stimuli presented to a BCM neuron. \(x^{(1)} = (\cos \pi/24, \sin \pi/24)\) is red and \(x^{(2)} = (\sin \pi/24, \cos \pi/24)\) is blue. Right: oscillatory response of the neuron over time to each of the two stimuli. The colors of the plots correspond to the colors of the stimuli. The initial conditions of \(w_1, w_2, and \theta\) lie in the interval \((0, 0.5)\), \(\tau_w = 100\), and \(\tau_\theta = 135\).](image)

We have observed that the response values, and thus the selectivity, of the neuron depend on the ratio of the sizes of the clusters. This also implies that in a two
stimuli case – like as studied in section 3.1 – the relative values of the responses
would then depend on the probability with which the stimuli were presented. Our
studies have also shown that When the threshold, \( \theta \) changes slower than the weights,
\( w \), the dynamics of the BCM neuron take on a different kind of behavior: the re-
sponses oscillate. fig. 5 shows results for a slightly modified version of the simulation
performed in section 3.1. Again the two stimuli are \( x^{(1)} = (\cos \pi/24, \sin \pi/24) \) and
\( x^{(2)} = (\sin \pi/24, \cos \pi/24) \) – respectively the red dot and the blue dot on the left
panel – and the initial conditions of \( w_1 \), \( w_2 \) and \( \theta \) lie in the interval (0, 0.5). Also
\( \text{Pr}[x(t) = x^{(1)}] = \text{Pr}[x(t) = x^{(2)}] = \frac{1}{2} \), however, \( \tau_w = 10 \) while \( \tau_\theta = 135 \).

As can be seen, the dynamics of the system does not yield a stable steady state.
It gives rise to an oscillation; though a stable oscillation. This behavior does not
lend itself well to selectivity. Let \( v_1 = w \cdot x^{(1)} \) be the response of the neuron to \( x^{(1)} \),
and \( v_2 = w \cdot x^{(2)} \) the response of the neuron to \( x^{(2)} \). Even though \( v_1 \) looks like it
is always higher than \( v_2 \), the overall ranges of \( v_1 \) and \( v_2 \) intersect. The oscillatory
behavior observed here has to do with the new choices of \( \tau_w = 10 \) and \( \tau_\theta = 135 \)
and will be studied further in section 4.

4. Steady state and stability

Although the scope of the present work does not include an indepth study of the
dynamical system of the BCM learning – which have been explored in other work
including Ref. 35 – we use this section to explore the plausible stability regimes of
the learning rule as it relates to the goal of this work. The dynamics of the BCM
neuron (eq. (2)) are stochastic in nature, since at each time step, the neuron ran-
domly receives one stimulus out of a set of stimuli. One way to gain more insight
into the nature of these dynamics is to study a mean-field deterministic approxi-
mation of the learning rule. If the rate of change of the stimuli is rapid compared
to that of the weights and threshold, then we can average over the fast timescale
to get a mean field or averaged model and then study this through the usual meth-
ods of dynamical systems. Consider a BCM neuron that receives a stimulus input
\( x \), stochastically from the set \{ \( x^{(1)} = (x_{11}, x_{12}, \ldots, x_{1n}), x^{(2)} = (x_{21}, x_{22}, \ldots, x_{2n}) \) \} such that
\( \text{Pr}[x(t) = x^{(1)}] = \rho \) and \( \text{Pr}[x(t) = x^{(2)}] = 1 - \rho \). A mean field equation
for the synaptic weights is

\[
\dot{w}_i = \rho x_i v_1 (v_1 - \theta) + (1 - \rho) x_i v_2 (v_2 - \theta), \quad i \in \{1, 2, \ldots, n\}
\]

Now let the responses to the two stimuli be \( v_1 = w \cdot x^{(1)} \) and \( v_2 = w \cdot x^{(2)} \). With
this, changes in the responses can be written as

\[
\begin{align*}
\dot{v}_1 &= x_{11} \dot{w}_1 + x_{12} \dot{w}_2 + \ldots + x_{1n} \dot{w}_n \\
\dot{v}_2 &= x_{21} \dot{w}_1 + x_{22} \dot{w}_2 + \ldots + x_{2n} \dot{w}_n
\end{align*}
\]  

(6)

So a mean field equation in terms of the responses is

\[
\begin{align*}
\tau_w \dot{v}_1 &= \rho x^{(1)} \cdot (v_1 - \theta) + (1 - \rho) x^{(2)} v_2 (v_2 - \theta) \\
\tau_w \dot{v}_2 &= \rho x^{(1)} \cdot (v_1 - \theta) + (1 - \rho) x^{(2)} \cdot (v_2 - \theta) \\
\tau_\theta \dot{\theta} &= \rho v_1^2 + (1 - \rho) v_2^2 - \theta
\end{align*}
\]  

(7)
We note that we could also write down the averaged equations in terms of the weights, but this form of the equations is much more cumbersome. We now look for equilibria and the stability of these equations. We note that if the two stimuli are not colinear and $\rho \in (0, 1)$, then $\dot{v}_{1,2} = 0$ if and only if $v_j(v_j - \theta) = 0$, $j \in \{1, 2\}$. Using the fact that at equilibrium, $\dot{v} = \rho v_1^2 + (1 - \rho)v_2^2$, we find:

$$
\begin{align*}
\dot{v}_1(v_1 - (\rho v_1^2 + (1 - \rho)v_2^2)) &= 0 \\
\dot{v}_2(v_2 - (\rho v_1^2 + (1 - \rho)v_2^2)) &= 0
\end{align*}
$$

(8)

which gives the fixed points

$$(v_1, v_2, \theta) = \left\{ (0, 0, 0), \left( \frac{1}{\rho}, 0, \frac{1}{\rho} \right), \left( 0, \frac{1}{1 - \rho}, \frac{1}{1 - \rho} \right), (1, 1, 1) \right\} \quad (9)$$

We will ignore the fixed points $(0, 0, 0)$ and $(1, 1, 1)$ as they are neither stable nor selective – and thus not useful to the task at hand. Ref. 4 and Ref. 35 discussed the stability of these unstable fixed points as they pertain to the original formulation. Ref. 6 and Ref. 24 gave a similar treatment to the objective function formulation. In the context of the main focus of this paper, the following theorem summarizes the steady state result detailed above.

**Theorem 1.** Assume that a BCM neuron is trained with a set of two points \{x(1), x(2)\}, and that the data points are presented with respective probabilities $\rho$ and $1 - \rho$. Let the responses of the neuron to $x(1)$ and $x(2)$ be $v_1$ and $v_2$ respectively. Then, at selective steady state, either $(v_1, v_2) = (1/\rho, 0)$ or $(v_1, v_2) = (0, 1/(1 - \rho))$.

As a side note, we observe that when $\rho = 1/2$, the selective fixed point is either $(2, 0, 2)$ or $(0, 2, 2)$ and corresponds to the results seen in fig. 2, fig. 4A, and fig. 4B. The plots in these figures suggest that the combination of parameters we employed in the simulations allow for stability of the fixed point in each case. Notice, however, that $\rho = 1/2$ also in fig. 5 but the plots show that the fixed point in this case is not stable. We proceed to explore the stability of these fixed points as a function of the angle between the stimuli, the probability factor of the stimuli, $\rho$, and the ratio of $\tau_0$ to $\tau_w$.

To simplify our calculations we translate the common tail of the vectors $x(1)$ and $x(2)$ so they have equal magnitudes, and assume – without loss of generality – that $x(1) \cdot x(1) = x(2) \cdot x(2) = 1$. To further simplify the calculations, we let $\tau = \tau_0/\tau_w$, $\beta = x(1) \cdot x(2)$, and $r = \rho/(1 - \rho)$. Note that $r \in (0, \infty)$ with $r = 1$ being the case of equal probability.

The Jacobian of eq. (7) is

$$
J = \begin{pmatrix}
\frac{(2\rho v - \rho \theta)}{\tau_w} & (2(1 - \rho)bv_2 - (1 - \rho)b\theta)/\tau_w & (-\rho v_1 - (1 - \rho)bv_2)/\tau_w \\
\frac{(2\rho b_1 - \rho b\theta)}{\tau_w} & (2(1 - \rho)v_2 - (1 - \rho)b\theta)/\tau_w & (-\rho b_1 - (1 - \rho)v_2)/\tau_w \\
\frac{(2\rho v_1)}{\tau_w} & (2(1 - \rho)v_2)/\tau_0 & -1/\tau_0
\end{pmatrix}.
$$
Thus to linearize the equation at $\mathbf{z}_1 \equiv \left( \frac{1}{\rho}, 0, \frac{1}{\rho} \right)$, we evaluate the above Jacobian at this point to obtain

$$J_{\mathbf{z}_1} = \begin{pmatrix} 1 - \beta/r & -1 \\ \beta & -1/r - \beta \\ -2/\tau & 0 & -1/\tau \end{pmatrix}. \tag{10}$$

where for simplicity, we rescale time and let $\tau_w = 1$. The characteristic polynomial becomes

$$p_{\mathbf{z}_1}(\lambda) = \det(\lambda I - J_{\mathbf{z}_1})$$

$$= \begin{vmatrix} \lambda - 1 & -\beta/r & 1 \\ \beta & \lambda + 1/r & \beta \\ -2/\tau & 0 & \lambda + 1/\tau \end{vmatrix} \tag{11}$$

$$= -2\{\beta^2/r - (\lambda + 1/r)\}/\tau + (\lambda + 1/\tau)\{(\lambda - 1)(\lambda + 1/r) + \beta^2/r\}$$

$$= \lambda^3 + A_1\lambda^2 + B_1\lambda + C_1$$

where

- $A_1 = (1/r + 1/\tau - 1)$
- $B_1 = (1/\tau - 1/r + \beta^2/r + 1/\tau r)$
- $C_1 = (-\beta^2/\tau r + 1/\tau r)$

Observe that if we assume that $0 < \tau < 1$, then we can be sure that $A > 0$. Also notice that since $\mathbf{x}^{(1)} \cdot \mathbf{x}^{(1)} = \mathbf{x}^{(2)} \cdot \mathbf{x}^{(2)} = 1$, we obtain that $\beta^2 < 1$ implying that $C_1 > 0$. The Routh Hurwitz criterion (Ref. 36) can be employed to provide more insight on the stability of $\mathbf{z}_1$. According to this criterion, the roots of the above characteristic polynomial all have negative real parts if and only if $A_1 > 0$, $C_1 > 0$ and $R_1 := A_1B_1 - C_1 > 0$. \textsuperscript{16} Thus if $0 < \tau < 1$, then the stability of $\mathbf{z}_1$ is guaranteed if and only if

$$R_1 := \frac{(r + \tau - \tau r)(r - \tau + \beta^2 + 1 + \tau \beta^2 - 1)}{\tau r} > 0 \tag{12}$$

Applying a similar treatment to $\mathbf{z}_2 \equiv \left( 0, \frac{1}{1-\rho}, \frac{1}{1-\rho} \right)$, we obtain the Jacobian

$$J_{\mathbf{z}_2} = \begin{pmatrix} -r & \beta & -\beta \\ -r\beta & 1 & -1 \\ 0 & 2/\tau & -1/\tau \end{pmatrix}. \tag{13}$$
The characteristic polynomial in this case is
\[ p_{z_2}(\lambda) = \text{det}(\lambda I - J_{z_2}) = (\lambda + r - \beta \lambda - 1) \begin{vmatrix} \lambda + r & -\beta & \beta \\ r\beta & \lambda - 1 & 1 \\ 0 & -2/\tau & \lambda + 1/\tau \end{vmatrix} = 2(\lambda + r - \beta \lambda - 1) \begin{vmatrix} \lambda + r & -\beta \lambda - 1 \end{vmatrix} \]
\[ = \lambda^3 + A_2\lambda^2 + B_2\lambda + C_2 \] (14)

where
\[ A_2 = (r + 1/\tau - 1) \]
\[ B_2 = (1/\tau - r + \beta^2 r + r/\tau) \]
\[ C_2 = (-r\beta^2/\tau + r/\tau) \]

Again we observe that the assumption of 0 < \( \tau < 1 \) will imply that \( A > 0 \); also since \( x^{(1)} \cdot x^{(1)} = x^{(2)} \cdot x^{(2)} = 1 \), we get that \( \beta^2 < 1 \) implying that \( C_2 > 0 \). To apply the Routh Hurwitz criterion, we note that \( A_2 > 0, C_2 > 0 \) and \( R^2 := A_2B_2 - C_2 > 0 \). Thus if 0 < \( \tau < 1 \), then \( z_2 \) is asymptotically stable if and only if
\[ R_2 := \frac{(r + \tau r - \tau)(1 - \tau r + \beta^2 \tau r + r) + r(\beta^2 - 1)}{\tau} > 0 \] (15)

Collating the discussion that led to eq. (12) and eq. (15), we obtain the following theorem about the stability of our selective steady state:

**Theorem 2.** Let \( \beta = x_1 \cdot x_2, \tau = \tau_0/\tau_w \) and \( r = \rho/(1 - \rho) \). Then the qualitative behaviors of the steady state solutions presented in theorem 1 can be summarized as follows:

- \((v_1, v_2) = (1/\rho, 0)\) is linearly asymptotically stable if and only if \( \tau < 1 \) and \((r + \tau - r\tau)(r - \tau + \beta^2 + 1) + \tau r(\beta^2 - 1) > 0 \)
- \((v_1, v_2) = (0, 1/(1 - \rho))\) is linearly asymptotically stable if and only if \( \tau < 1 \) and \((r + \tau r - \tau)(1 - \tau r + \beta^2 \tau r + r) + r(\beta^2 - 1) > 0 \)

5. **A single-neuron BCM clustering algorithm**

We propose an algorithm that makes use of the selectivity property of the BCM neuron to cluster a dataset. The algorithm for this task will involve the variables \( X, n, d, v, w, \theta, \tau_w, \tau_0, R, \kappa, \) and \( \delta \) characterized as follows:

1. \( X \) is an \( n \times d \) matrix containing the data set.
2. \( n \) is the number of data points in the data set.
3. \( d \) is the number of attributes of each data point i.e the dimension of the dataset.
Cluster-Detecting Property of the BCM Neuron

(4) \( v \) is the activity of the neuron.
(5) \( w \) is a \( d \)-dimensional vector of the synaptic weights of the neuron. To avoid convergence to a degenerate fixed point, \( w \) should not be initialized with the zero vector.
(6) \( \theta \) is the activity threshold of the neuron.
(7) \( R \) is an \( n \)-dimensional vector. The element \( R_i \) corresponds to the response of the neuron to the \( i \)th data point of \( X \).
(8) \( \tau_w \) is the time-scale factor for \( w \).
(9) \( \tau_\theta \) is the time-scale factor for \( \theta \). As has been shown in section 4 and section 3, steady state stability is guaranteed if \( \tau_\theta / \tau_w < 1 \).
(10) \( \text{numItr} \) is the number of iterations during which the neuron is trained.
(11) \( \kappa \) and \( \delta \) are cluster detection cutoff factors. \( \delta \) is an expected average difference in the neuronal responses between two data points in the same cluster. This is usually very small (say, in the order of 0.1). \( \kappa \delta \) is the expected minimum difference in the neuronal responses between two data points in two different clusters, implying that \( \kappa \) should be chosen in a such a way that \( \kappa \delta \) is large enough to reflect this. However, as can be deduced in the algorithms that follow, if \( \kappa \) is too large, points belonging to different clusters may be grouped together. Other than by experimentation, there is currently no hard rule in determining the value of \( \kappa \) and \( \delta \).

The mechanism of the proposed single-neuron BCM clustering algorithm (algorithm 2) can be described as \textit{detect and eliminate}. This means that it trains the BCM neuron with the dataset, detects the high-response data points, designate these points as a cluster, and eliminate them from the dataset. It then repeats this procedure until the dataset is empty. The algorithm can easily be modified to accept a pre-defined number of desired clusters.

1: Initialize \( v = 0, \theta = 0, w = \) a non-zero vector
2: \textbf{for} \( \text{itr} = 1 \) to \( \text{numItr} \) \textbf{do}
3: \hspace{1em} \( x = \) a random row of \( X \)
4: \hspace{1em} \( v = x w^T \)
5: \hspace{1em} \( w = w + \frac{1}{\tau_w} v (v - \theta) x \)
6: \hspace{1em} \( \theta = \theta + \frac{1}{\tau_\theta} (v^2 - \theta) \)
7: \textbf{end for}
8: \textbf{return} \( R = X w^T \)

\textbf{Algorithm 1:} Using a data set \( X \) to train a BCM neuron

5.1. \textit{Performance}

The primary goal of the present work is to analyze the cluster-detection property of the BCM neuron – rather than to develop and rigorously test a novel clustering
while $X$ is not empty do
1. Train neuron by running algorithm 1 and computing $R$;
2. $S = \text{sort}(R)$ in descending order;
3. $\delta = \text{very small positive number}$;
4. $\kappa = \text{positive number greater than 1}$;
5. if $\text{length}(S) == 1$ then
6. $X$ contains just one row (i.e. one data point);
7. Designate this row as a cluster, and eliminate it from $X$;
8. else if $\text{length}(S) == 2$ then
9. $X$ contains just two rows (i.e. two data points);
10. if $S_1 - S_2 > \kappa \delta$ then
11. Designate row 1 as a cluster and row 2 as another clusters, and
12. eliminate both rows from $X$;
13. else
14. Designate the two rows as a cluster, and eliminate both rows from $X$;
15. end if
16. else
17. $j = \text{the minimum integer less than length}(S)$ such that
18. $S_j - S_{j+1} \geq \kappa(S_{j+1} - S_{j+2})$;
19. If no such $j$ exists, then assign $j = \text{length}(S)$;
20. Assign $C = \text{a subset } \{X_i\}$ of the rows of $X$ whose responses $\{R_i\}$ are
21. such that each $R_i \geq S_j$;
22. Designate $C$ as a cluster, and eliminate the rows in $C$ from $X$;
23. end if
24. end while

**Algorithm 2:** Using a BCM neuron to cluster a dataset $X$

---

algorithm. For this reason we choose datasets that will properly demonstrate this goal. We concede that future work will include fine-tuning this preliminary clustering algorithm to account for probability distribution and other relevant factors. With that being said, we proceed to apply algorithm 2 on the some select datasets. In each simulation, $\tau_w = 10$, $\tau_\theta = 1$, $numItr = 10000$, $\kappa = 10$, $\delta = 0.05$, and $w$ is initialized with numbers in the interval $(0, 0.3)$.

- **Artificially generated dataset:** A dataset is said to be linearly separable if for each cluster in the data set, there exists a hyperplane that can be used to separate the cluster from the remainder of the dataset. The datasets in fig. 6 are generated in the same manner that the ones in fig. 4 were generated. fig. 6A and 6B show the best performances of algorithm 2 on two linearly separable data sets. The numbers on the plots show the order in which the the clusters were detected. fig. 6C shows the performance of the algorithm on a data set that is not linearly separable. In each case we specified the number of desired
clusters.

Fig. 6. (A)-(B) Best performances of algorithm 2 on two linearly separable data sets (C) Performance of the algorithm on a data set that is not linearly separable. The numbers on the plots show the order in which the clusters were detected.

• Iris data set: This data set was introduced by Sir Ronald Fisher in 1936\textsuperscript{15} and can be found on the UCI machine learning repository \textsuperscript{1}. The set contains 150 data points on three species of the Iris flower comprising 50 samples from the Iris setosa species, 50 from Iris virginica, and 50 from Iris versicolor. It measures (in cm) the following four attributes for each sample: length of the sepal, the width of the sepal, the length of the petal, and the width of the petals. This dataset is hardly analyzed by clustering since there are only two obvious clusters (within three groups): one of the clusters contains Iris setosa, while the other cluster contains both Iris virginica and Iris versicolor and is not separable without the species information Fisher used. Fig. 7A shows a 3-d projection of the iris data points. The cyan points are samples of iris setosa, the purple points are samples of iris virginica, and the yellow points are samples of iris versicolor. Fig. 7B shows a performance of algorithm 2 that comes very close to mirroring the species clusters of the dataset. Again, the numbers on the
plots show the order in which the clusters were detected and the number of desired clusters, 3, was pre-defined.

Fig. 7. (A) The cyan points are samples of iris setosa, the purple points are samples of iris virginica, and the yellow points are samples of iris versicolor (B) A performance of algorithm 2 that comes very close to mirroring the species clusters of the dataset. The numbers on the plots show the order in which the clusters were detected.

- **Cardiotocography dataset**: This data set consists of measurements of fetal heart rate (FHR) and uterine contraction (UC) features on cardiotocograms classified by expert obstetricians. This dataset can be found on the UCI machine learning repository\(^1\). The dataset consists of 2126 samples and each sample has 23 attributes. With respect to morphological patterns, each sample in the data set can be placed into one of ten groups A, B, C, D, ... J. Each of the samples is already pre-classified to be in one of these groups. We set algorithm 2 to detect exactly 10 clusters in order to see how the clusters compare to the original 10 groups in the dataset. We first choose a representative centroid for each of the 10 groups A, B, C, D, ... J. This representative centroid is the point in the group that is closest to the mean of the group. Before clustering, we associate each data point with a representative centroid from original grouping. We thus consider a point to be *correctly clustered* if after the clustering it belongs in the same cluster as the associated representative centroid. Over 100 runs of algorithm 2, we found the average percentage of correctly clustered points to be 79.23%.

To further evaluate the performance of algorithm 2 we compute the Davies-Bouldin (DB) index\(^11\) for the clusters produced by the algorithm, as well as the DB index for clusters produced by a few other popular clustering algorithms. The DB index takes into consideration a combination the similarity of points within the clusters and the dissimilarity between the clusters and is calculated in the following...
Table 1. Mean DB indices (over 100 runs) for different clustering algorithms

<table>
<thead>
<tr>
<th>Data set</th>
<th>BCM (algorithm 2)</th>
<th>k-means</th>
<th>EM</th>
<th>CLINK</th>
</tr>
</thead>
<tbody>
<tr>
<td>figure 6A dataset (3 clusters)</td>
<td>0.3001</td>
<td>0.3228</td>
<td>0.3180</td>
<td>0.3110</td>
</tr>
<tr>
<td>figure 6B dataset (4 clusters)</td>
<td>0.3009</td>
<td>0.3100</td>
<td>0.3013</td>
<td>0.2910</td>
</tr>
<tr>
<td>figure 6C dataset (3 clusters)</td>
<td>0.3611</td>
<td>0.4200</td>
<td>0.3991</td>
<td>0.4710</td>
</tr>
<tr>
<td>Iris dataset (3 clusters)</td>
<td>0.3910</td>
<td>0.4012</td>
<td>0.4024</td>
<td>0.3211</td>
</tr>
<tr>
<td>Cardiotocography dataset (10 clusters)</td>
<td>0.3411</td>
<td>0.3510</td>
<td>0.3521</td>
<td>0.4437</td>
</tr>
</tbody>
</table>

where

\[ DB = \frac{1}{n} \sum_{i=1}^{n} \max_{i \neq j} \left( \frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right) \]

where

- \( n \) is the number of clusters.
- \( c_x \) is the centroid of cluster \( x \).
- \( \sigma_x \) is the average distance of all elements in cluster \( x \) to centroid \( c_x \).
- \( d(c_i, c_j) \) is the distance between centroids \( c_i \) and \( c_j \).

In a clustering result with high intra-cluster similarity and low inter-cluster similarity, the numerator of the inside of each term of the summation above should be less than the denominator. Thus, a reasonable DB index should be less than one. In addition, the algorithm that yields the best clustering result should have the lowest DB index.

Table 1 shows the DB indices for different algorithms on the datasets described above. Each algorithm is adjusted to yield the same number of clusters associated with each dataset, and each number is the average DB index over 100 runs. As can be seen, algorithm 2 performs very well when compared with the other clustering algorithms. The bolded numbers highlight cases where, of all the algorithms, algorithm 2 gives the best results. As expected the algorithm performs best when the clusters are clearly linearly separable.

5.2. Convergence and computational complexity

The BCM clustering algorithm uses Euler’s method to solve the system of ordinary differential equations in the BCM learning rule. The convergence rate of Euler’s method can be analytically determined for Lipschitz continuous functions,
and pretty difficult to determine for functions that are not Lipschitz continuous, especially when stochasticity is involved, like in algorithm 2. Even though the BCM learning rule is not Lipschitz continuous and determining its convergence rate is not within the scope of the present study, our experimental simulations (see fig. 4) show that as long as $\tau_0/\tau_w < 1$, the learning rule converges in a reasonable amount of time.

In terms of computational complexity, the parts of algorithm 2 that contribute in determining its computational complexity are the for-loop that starts on line 2, and the matrix-vector multiplication in line 8. The for-loop runs a constant number of times ($numItr$) which does not depend on the size of the data. Line 5 is the dot product of two $d$-dimensional vectors, and has a complexity of $O(d)$. Each of lines 5 and 6 has a complexity of $O(1)$ since the dominant operation in each of them is the multiplication of a vector by a constant. Thus the complexity of lines 2 to 7 of the algorithm is $O(d)$. Line 8 is the multiplication of an $n \times d$ matrix by a $d-$dimensional vector, which has a complexity of $O(nd)$. Therefore the overall complexity algorithm 1 is $O(nd)$.

In terms of computational complexity, the major parts of the algorithm 2 are line 2, line 3, and the if-else block (lines 6-20). The contributions of these major parts, can be enumerated as follows:

- As explained above, Line 2 has a complexity of $O(nd)$.
- Line 3 can be done with a merge sort, which has a complexity of $O(n \log n)$.
- The most computationally complex part of the if-else block is lines 17 to 20. Line 17 is a sequential comparison of consecutive elements of an $n-$dimensional vector, and has a complexity of $O(n)$ . Line 18 also has a complexity of $O(n)$ since it is the sequential comparison of the elements of an $n-$dimensional vector to a threshold. Line 19 is the deletion or flagging of at most $n$ rows in a matrix, and has a complexity of $O(n)$.

If $m$ is the number of clusters detected, the while loop that starts at line 1 and ends at line 21 runs exactly $m$ times . In the worst case scenario, which is very unlikely, $m = n$. Otherwise $m < n$. Thus the overall complexity of the algorithm 2 is $O(m(n \log n + nd))$ where $m$ is the number of clusters, $d$ is the number of attributes of each data point, and $n$ is the number of points in the data set. This is generally better than the complexity of the other three algorithms we compared with algorithm 2. Each of EM and k-means clustering algorithms has a best case complexity of $O(n^{nd+2} \log n)$, and the complete linkage hierarchical clustering algorithm has a complexity of $O(n^3)$.5

Table 2 show the mean CPU time, in seconds, (over 100 runs) of each clustering algorithm on the data sets already discussed. As can be seen, the BCM clustering algorithm is faster than the other algorithms. On the cardiotocography data set for instance, the algorithm took only about 80% of the CPU time of k-means.
Table 2. Mean CPU time, in seconds, (over 100 runs) for different clustering algorithms

<table>
<thead>
<tr>
<th>Data set</th>
<th>BCM (algorithm 2)</th>
<th>k-means</th>
<th>EM</th>
<th>CLINK</th>
</tr>
</thead>
<tbody>
<tr>
<td>figure 6A dataset</td>
<td>0.4108</td>
<td>0.5004</td>
<td>0.5891</td>
<td>0.5021</td>
</tr>
<tr>
<td>figure 6B dataset</td>
<td>0.5108</td>
<td>0.5704</td>
<td>0.6891</td>
<td>0.6021</td>
</tr>
<tr>
<td>Iris data set</td>
<td>0.4113</td>
<td>0.6100</td>
<td>0.8711</td>
<td>0.4364</td>
</tr>
<tr>
<td>Cardiotocography data set</td>
<td>5.8753</td>
<td>7.2953</td>
<td>7.7613</td>
<td>7.8553</td>
</tr>
</tbody>
</table>

6. Discussion

We have explored the ability of the BCM neuron to detect clusters in a dataset. Our main approach involved treating data points in a dataset as stimuli presented to a BCM neuron. We have shown that in this context, the following factors contribute to the stability of the selective fixed points of the BCM learning rule for a single neuron: the amplitudes of the competing stimuli; the probabilities with which the stimuli are presented; and the time scale of the sliding threshold. Similar results would be possible if the quadratic term, \( v^2 \), in the second line of equation 2 were replaced with \( v^p \), \( p > 2 \), since the original formulation in Ref. 4 suggests that the fixed point structures are preserved for any positive value of \( p \).

Our analytical results in section 3 and section 4 leaned heavily on scenarios where the neuron is selecting between two data points or two clusters. We found that this is sufficient since our overall approach is for the neuron to detect one cluster at a time. Thus if there are more than two clusters, the analyses in these sections will still be apply if one uses a one-vs.-rest approach. We also considered other approaches but decided not include them in this paper. One such approach is to use a small network of coupled BCM neuron, with the number of neurons corresponding to the number of desired clusters. To make such a network receive stimulus patterns from a common set and work for the task at hand, it is important to incorporate a mechanism for competitive selectivity within the network. A mechanism of this sort, found in visual processes (and also in tactile, auditory, and olfactory processing) is called lateral inhibition. During lateral inhibition, an excited neuron reduces the activity of its neighbors by disabling the spreading of action potentials to neighboring neurons in the lateral direction. It is thus expected that, at the end, each neuron would select one unique cluster.

Another approach we considered involves observing the response of all the clusters at once and cluster-labeling points based on where their response lie among the subset of responses. For instance, fig. 8 shows the distribution of the responses of a BCM neuron to three separable data clusters generated in a fashion similar to how the clusters of fig. 4 were generated. We however found this approach to not be robust enough when the number of clusters is high.

The model neuron we used in this paper has been assumed to have linear re-
response properties, which may be seen as oversimplified. However, this approach is sufficient for the scope of our present study. The introduction of a nonlinear transfer function to the BCM learning rule has been addressed by Ref. 24. In their formulation, the learning rule is derived as a gradient descent rule on an objective function that is cubic in the nonlinear response. Our decision to use linear neurons is motivated by the accessibility to formal analysis.

The clustering algorithm proposed in section 5 (algorithm 2) does not require specification of the desired number of clusters (though if there is need to to this, it can easily be achieved by simply replacing the ‘while’ loop with a ‘for’ loop). In this regard, the algorithm is more appealing than algorithms like k-means and EM clustering algorithms which are computationally more complex, and do require the user to specify the number of clusters beforehand. Nevertheless algorithm 2 has a few limitations. First, it only deals with data with numerical values and it does not handle missing attributes. Thus, a future work should gear towards improving the algorithm in such a way that it deals with data points with missing attributes, and at the same time handle data sets with non-continuous values. To improve the robustness of the algorithm, it may be useful to incorporate other distance measures apart from the Euclidean distance measure.

References
4. E. L. Bienstock, L. Cooper and P. Munro, Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex, Journal
18. R. Johansson and A. B. Vallbo, Tactile sensory coding in the glabrous skin of the
33. A. Poljovka and L. Benuskova, Pattern classification with the bcm neural network, *Unpublished*.