BCM Learning Rule

Report for the final project of the course "Neural Information Processing" in SoSe by Julia Hattendorf.

Introduction

- BCM rule introduced to account for "striking dependence of the sharpness of orientation selectivity on the visual env" https://www.researchgate.net/publication/2308452_Effect_of_Binocular_Cortical_Misalignment_on_Ocular_Dominance_and_Orientation_Selectivity

Bienenstock–Cooper–Munro

The Model

The forward pass is the usual weighted sum:

$$y = \sum_i w_i x_i$$

The BCM rule differs to hebbian learning in that the update of the weight is not a product of the presynaptic and postsynaptic activtiy but of the presynaptic activity, x, and a modification function, ϕ, of the the postsynaptic activity, y, and a threshold θ. [^1] Additionally, the authors allow the option for a weight decay term $-\epsilon w_i$ with a variable of decay ϵ. For the ith weight, the update looks like this:

$$\frac{dw_i}{dt} = \phi(y)x_i - \epsilon w_i$$

The modification function and its threshold, θ, make sure that the weight adapation relates to the postsynaptic activity: If the activity of a neuron low $(y < \theta)$ then its weight update will be negative. Vice versa, if it is higher $(y > \theta)$ the weight update will be positive. The weights should thus adapt to favor some inputs over others, so the rule induces competition between inputs.

The threshold itself also adapts based on the postsynaptic activity. So if the activity rises or falls so will the threshold. Ideally, the threshold would then prevent runaway activation.

In the orignal paper the threshold is calculated with the temporal average value, $E(\cdot)$, of the activity y and a positive constant y_o. The average value is additionally taken to the power of another positive constant p.

$$\theta = E^p[(y/y_o)]$$

Note that the authors : "The time average is meant to be taken over a period T preceding t much longer than the membrane time con- stant T so that E(t) evolves on a much slower time scale than c(t). "

note : postsynaptic firing rate

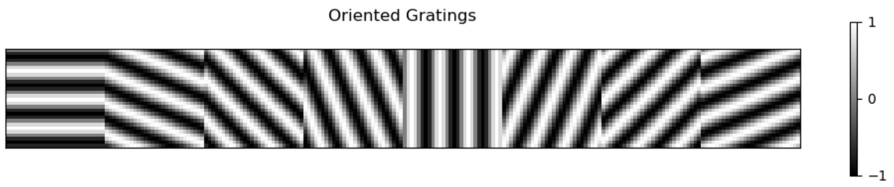
Interestingly, after this threshold model was introduced, biological evidence for it could be found in several brain regions (see REFS 94–101 for example), https://brabeeba.github.io/neuralReadingGroup/cooper.pdf

[^1]: Variable names from the original paper have been adjusted to normal machine learning conventions for readability.

Implementation & Problems

Dataset

For the dataset I chose synthetic oriented gratings as input. I then chose to construct an MNIST-like dataset, with 28x28 pxl per image and to repeat the orientations to have 400 samples in total. The dataset then had the shape (400, 784).



8 orientations evenly spaced over 180°.

Implementation Choices

I implemented the BCM learning rule in a class with Pytorch tensors. This allows for easy access of hyperparameters and traces. As is convention, I also implemented the update to be calculated on data batches (SGD-like).

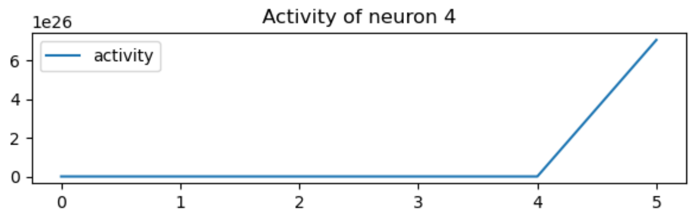
For the threshold, instead of a temporal average, I calculated the average of output activtiy per neuron over the input batch (using spatial instead of temporal average is discussed in Intrator & Cooper, (1992)).

For the modification function some sources state that the original functon used in the paper is $\phi = y(y - \theta)$ (Udeigwe et al., 2017) (Blais & Cooper, 2008). Altough, I could not confirm this with the paper I did chose it as a starting point.

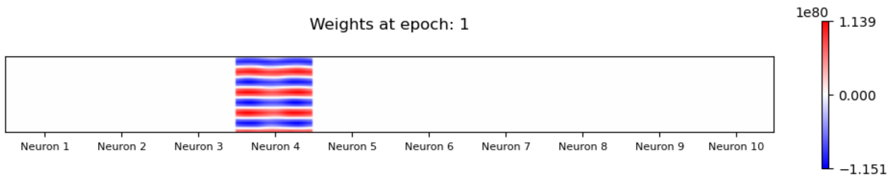
Problem: Runaway Activation

The BCM rule is supposed to solve runaway activity in hebbian learning but when I started implementing this rule, I had a big problem: runaway activation. When using the function $\phi = y(y - \theta)$ it is quite apparent that there is no upper bound to modifications and one can only rely on the threshold.

Running for n_epochs = 1 n_units = 10 batch_size = 100:



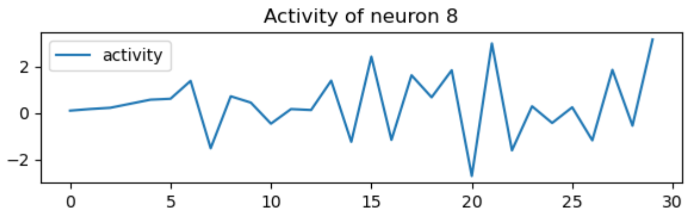
The activity of a single neuron while training.



The weights of all neuros after training.

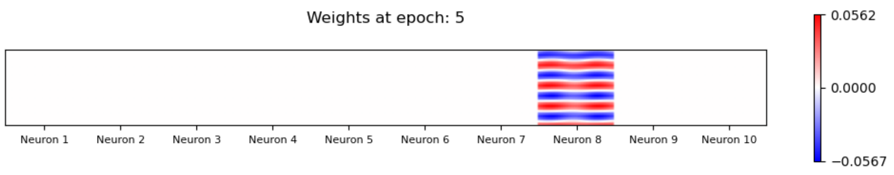
Solution 1: Weight normalization

The simplest fix seems to be to simply normalize the weight in every update.



The activity of a single neuron while training.

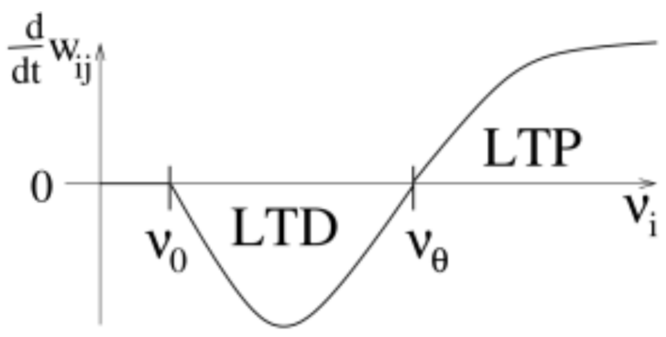
While that does help with the runaway activation problem, we still have the "snowball" problem of one neuron dominating:

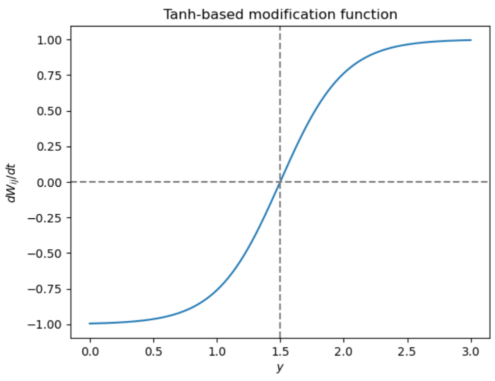


The weights of all neuros after training.

Solution 2: $\phi \rightarrow tanh$

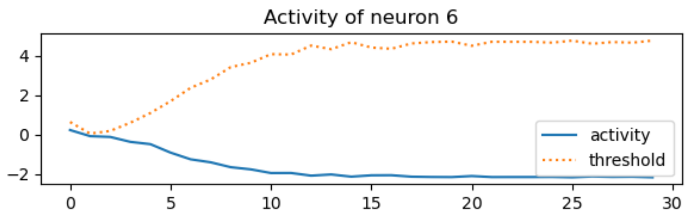
Since some sources show the modification funtion with a sigmoid-ish trajectory for $y > \theta$, I tried using a $tanh$-based function.



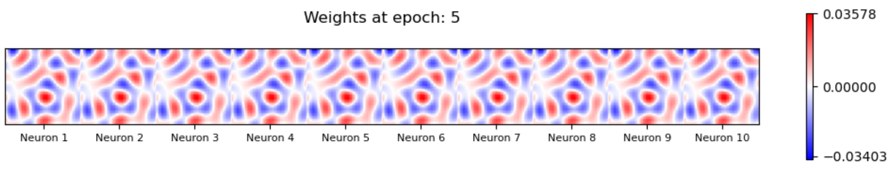


On the left the modification function according to the book Neuronal Dynamics by Gerstner et al. (2014). On the right my own variant $phi = tanh(2 * (y - \theta))$.

This also helped with the runaway problem but did not give me promising resutls in terms of feature extraction.



The activity and threshold of a single neuron while training.



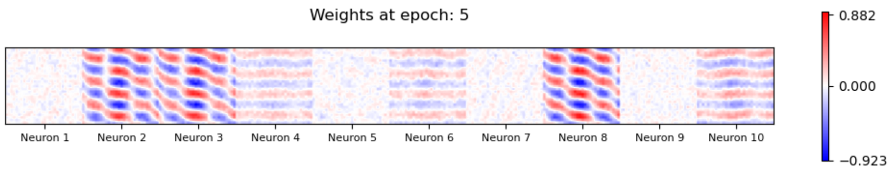
The weights of all neuros after training.

Solution 3: Curti implementation

In order to find another solution, I looked at the implemenation for the paper (Squadrani et al., 2022) that can be found on github account of Nico Curti. They use the implementation proposed by Law and Cooper in 1994. The differences to the original BCM imlplementation are:

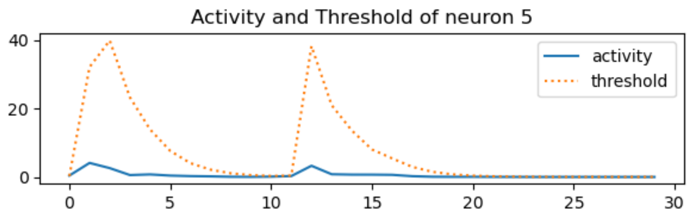
- adding ReLU as activation function: $y = ReLU(\sum_i w_i x_i)$
- calculating the threshold with the moving average: $\theta_t = \gamma\theta_{t-1} + (1-\gamma)\langle y^2 \rangle_{b_t}$ where $\langle \cdot \rangle_{b_t}$ is the average over the batch of training patterns
- dividing the weight update by the threshold: $\frac{dw_i}{dt} = \frac{y(y-\theta)x_i}{\theta}$

I found that no single of those differences is enough to ensure stability, but that all three are needed. After implementing all these differences, the form of the weight is improved and the orientations are picked up:



The weights of all neuros after training.

Additionally, the realtionship of threshold and activity are clearer. Below one can clearly see how spikes in output activity are picked up by the threshold and how the activity is regulated down:

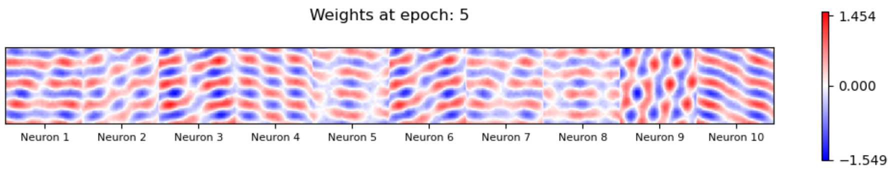


The activity and threshold of a single neuron while training.

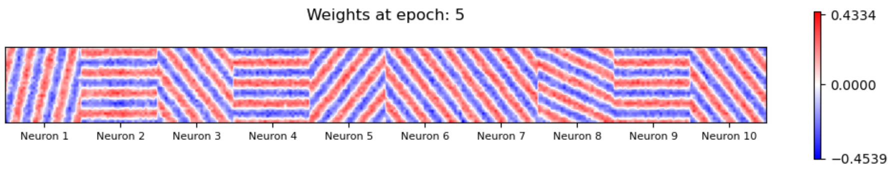
Timescales

To add another layer of complexity to this implementation, I added seperate timescales for the change of threshold, τ_θ, and the weight adaption, τ_w. To find an intuition for good values, I consulted the paper by Udeigwe et al. (2017), where the authors examined the dynamical properties for different values of τ_θ and τ_w. From the paper I gathered that $\tau_\theta > \tau_w$.

I unfortunately could not set up a proper gridsearch for these parameters due to time constraints and manually tried out a few combinations:



Weights for tau_th = 0.1 and tau_w = 0.05



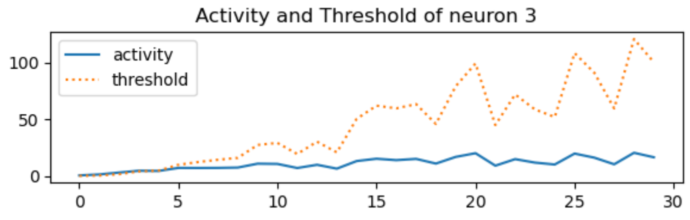
Weights for tau_th = 0.1 and tau_w = 0.001

The last combination of $\tau_\theta = 0.1$ and $\tau_w = 0.001$ is the one I will be using from now on for the analysis.

Results and Analysis

Threshold preventing runaway activity

The threshold is successfully regulting the activity. Looking at t=20 and t=25, the spike in activation is picked up by the threshold and the activation in the next step is reduced.



The activity and threshold of a single neuron while training.

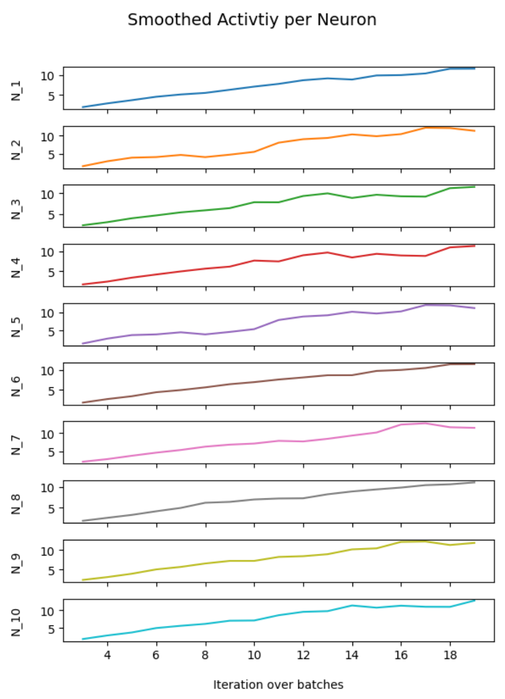
Selectivity increases with learning

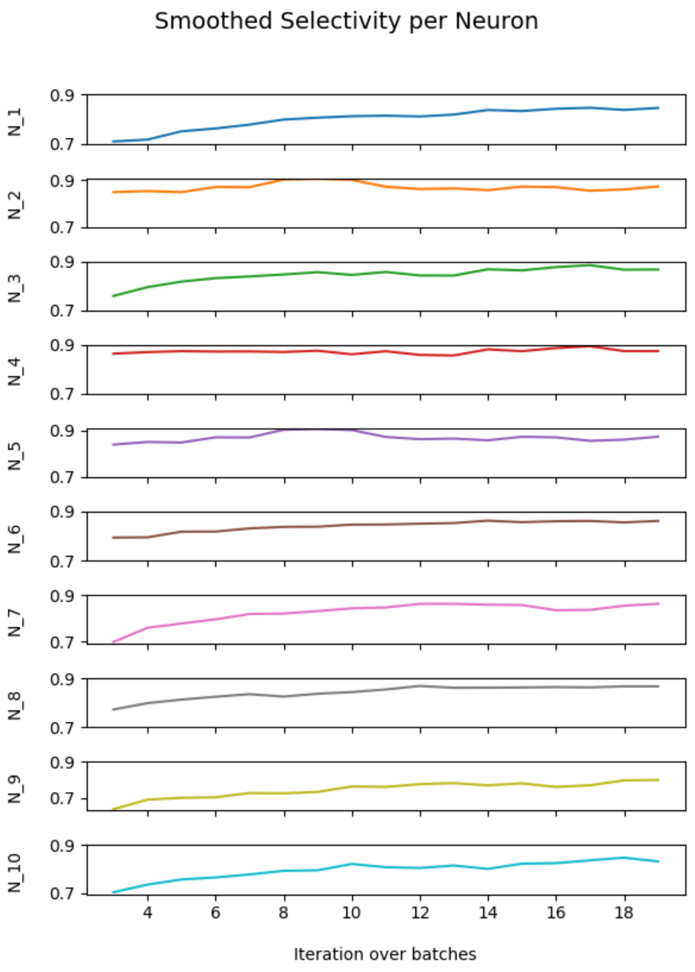
To have some measure for the "performance" of a neuron, I chose selectivity, i.e. how a neuron responds to certain stimuli compared to others. The general index for selectivity as proposed by Bienenstock, Cooper and Munro (Eq. 3, p. 33):

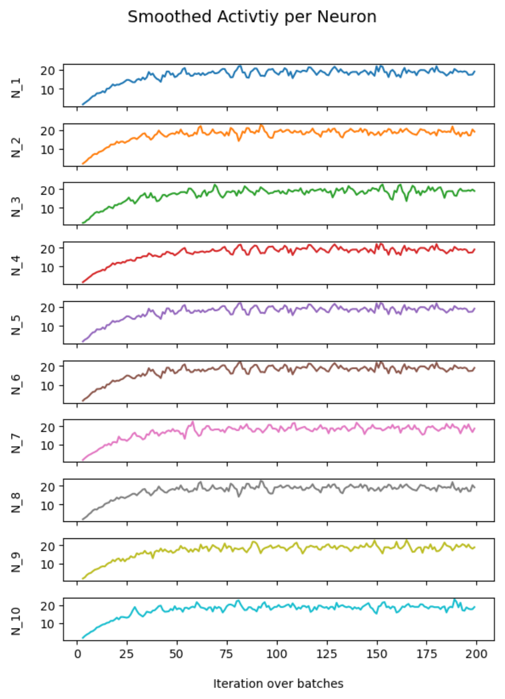
$$Sel_d(N) = 1 - \frac{\text{mean response of N with respect to d}}{\text{max response of N with respect to d}}$$

Stability

When looking at the activtiy of the neurons for 5 epochs (left) it looks like the runaway problem might still be there. However, training the neurons for 50 epochs (right) showed that the activity eventually converges. Both were trained with a batchsize of 100.







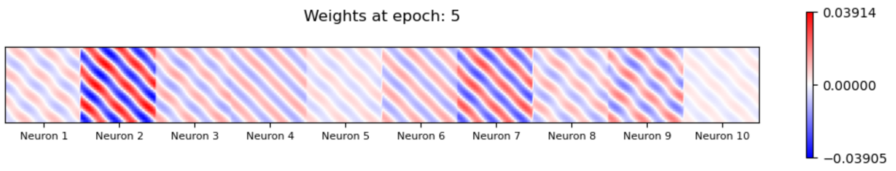
The activity of the neurons converge. Left trained for 5 epochs, right for 50 epochs.

Comparison

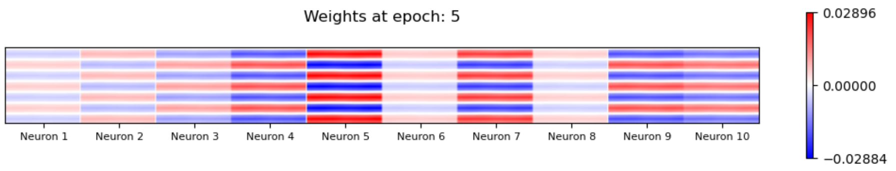
eta = 0.05 # learning rate

Hebb with weight normalization

Training the neurons with the hebb rule seems to lead to the same input image being picked up across all neurons. In different runs, different images are dominant:



The weights of all neuros after training with hebb.

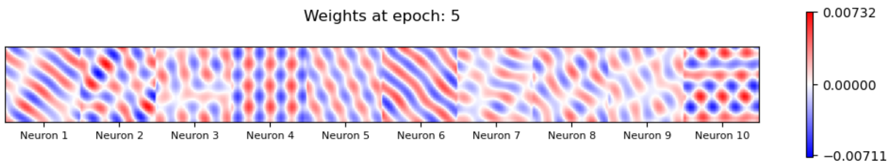


The weights of all neuros after training with hebb.

Oja

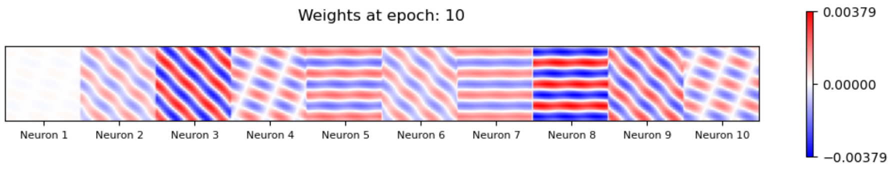
With Oja's rule I needed to train the network a little longer to achieve similar results to hebb and BCM.

For comparison, here is Oja's rule with 5 epochs:

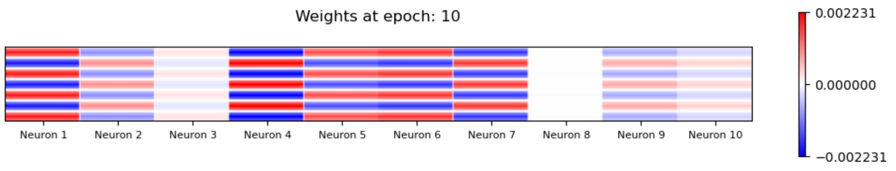


The weights of all neuros after training with Oja's rule.

And here for 10 epochs:



The weights of all neuros after training with Oja's rule.

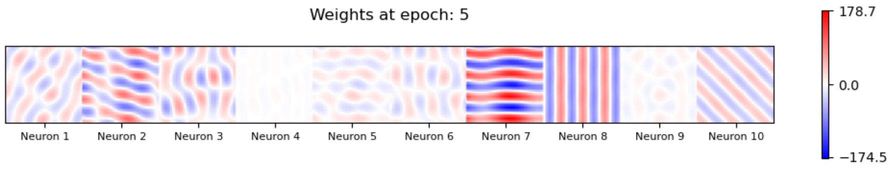


The weights of all neuros after training with Oja's rule.

Future ideas

WTA & BCM

Since BCM induces competition on an input pattern level and WTA induces competition on a neuron level, it makes sense to try and merge them together and see what happens. I attempted this, but am not quite satisfied with the results. Below are the weights trained with this combination and τ_θ = 1 and τ_w = 0.5.



The weights of all neuros after training with a WTA & BCM combination.

Further experiments would be needed here.

Investigating Timescales

My implementation has been based on the assumption that the "homeostatic" timescale, i.e. τ_θ is faster than the "learning" time scale, i.e. τ_w. I gathered this assumption from the work of Udeigwe et al. (2017). However, the authors also mention that this finding seems to be somewhat contradictory to the findings gathered from biological experiments, where the process of homeostatis seems to be slower than the process of learning. The authors also mention that there may be slow and fast homeostatic processes. It would be interesting to investigate this dilemma further.

Activation functions

The original model does not provide an activation function for the postsynaptic activtiy.

Research has showed that a good activation function should be:

- nonlinear "ot achieve nontrivial results" https://pmc.ncbi.nlm.nih.gov/articles/PMC9141587/
- positive for biological interpretation https://pmc.ncbi.nlm.nih.gov/articles/PMC9141587/
- function should be non symmetrical: https://www.researchgate.net/publication/2308452_Effect_of_Binocular_Cortical_Misalignment_on_Ocular_Dominance_and_Orientation_Selectivity

Often the sigmoid function is used as an activation https://pmc.ncbi.nlm.nih.gov/articles/PMC9141587/ Law and Cooper http://www.scholarpedia.org/article/BCM_theory

- Relu performs well in deep neural network learning so they chose RELU (Squadrani et al., 2022)

Effect of Inhibitory Synapse

show effect of inhib synapses by setting negative values to zero, selectivity should drop

- monocular versus binocular deprivation

Lit

Gerstner, W., Kistler, W. M., Naud, R., & Paninski, L. (2014). Neuronal dynamics: From single neurons to networks and models of cognition. Cambridge University Press.

https://neuronaldynamics.epfl.ch/online/Ch19.S2.html

Intrator, N., & Cooper, L. N. (1992). Objective function formulation of the BCM theory of visual cortical plasticity: Statistical connections, stability conditions. Neural Networks, 5(1), 3-17.

https://www.sciencedirect.com/science/article/pii/S0893608005800036

Squadrani, L., Curti, N., Giampieri, E., Remondini, D., Blais, B., & Castellani, G. (2022). Effectiveness of Biologically Inspired Neural Network Models in Learning and Patterns Memorization. Entropy, 24(5), 682.

https://pmc.ncbi.nlm.nih.gov/articles/PMC9141587/

Implementation on Nico Curti's github: https://github.com/Nico-Curti/plasticity/blob/main/plasticity/model/bcm.py

Udeigwe, L. C., Munro, P. W., & Ermentrout, G. B. (2017). Emergent dynamical properties of the BCM learning rule. The Journal of Mathematical Neuroscience, 7, 1-32.

https://mathematical-neuroscience.springeropen.com/articles/10.1186/s13408-017-0044-6

Blais, B. S., & Cooper, L. (2008). BCM theory. Scholarpedia, 3(3), 1570. http://www.scholarpedia.org/article/BCM_theory

Resources:

- http://www.scholarpedia.org/article/BCM_theory
- https://neuronaldynamics.epfl.ch/online/Ch19.S2.html
- https://www.researchgate.net/publication/2308452_Effect_of_Binocular_Cortical_Misalignment_on_Ocular_Dominance_and_Orientation_Selectivity
- https://pmc.ncbi.nlm.nih.gov/articles/PMC9141587/
- https://github.com/Nico-Curti/plasticity
- https://github.com/Nico-Curti/plasticity/blob/main/plasticity/source/bcm.pyx
- https://github.com/Nico-Curti/plasticity/blob/main/plasticity/model/bcm.py