# BCM Learning Rule

Report for the final project of the course "Neural Information Processing" in SoSe 2025.

## 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

## The Model

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, $\phi$, of the the postsynaptic activity, $y$, and a threshold $\theta$. [^1] Additionally, the authors allow the option for a weight decay term $-\epsilon w_i$ with a variable of decay $\epsilon$. 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, $\theta$, 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

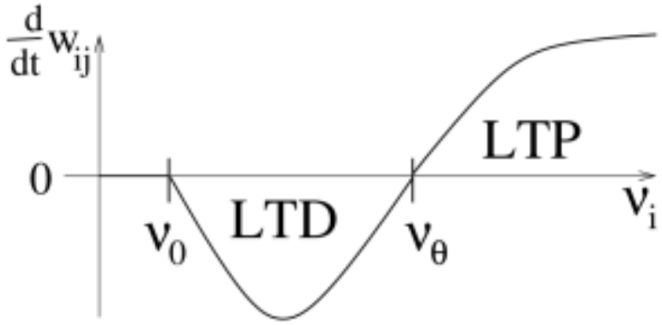
Interestingly, after this threshold model was introduced, biological evidence for it could be found in several brain regions (see REFS 94–101 for example), https://brabeeba.github.io/neuralReadingGroup/cooper.pdf

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

**Modification function $\phi$**

It is a bit unclear which modification function was used in the original paper. While some sources state that it was $\phi = y(y - \theta)$ [https://mathematical-neuroscience.springeropen.com/articles/10.1186/s13408-017-0044-

6 , scholarpedia], I could not confirm this. Additionally, the modification function is also often displayed as a combination of a sinus and a sigmoid (Intrator & Cooper, 1992):




Modification function according to the book Neuronal Dynamics by Gerstner et al. (2014)

From the explanation in the paper "The response of favored patterns grows until the mean response is high enough and the state stabilizes" (p. 36) I would assume

**Sliding Modification Threshold $\theta$**

Instead of a temporal average, using the spatial average has become popular https://www.sciencedirect.com/science/article/pii/S0893608005800036

Aside from that calculating the treshold with a moving average

$$\theta_t = \gamma\theta_{t-1} + (1-\gamma)\langle z^2\rangle_{b_t}$$

(Squadrani et al., 2022)

and a first order low-pass filter (Udeigwe et al., 2017) have been introduced

$$\tau_\theta \frac{d\theta}{dt} = (v^2 - \theta)$$

**Activation Function**

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

Research has showed that a good activation function should be:

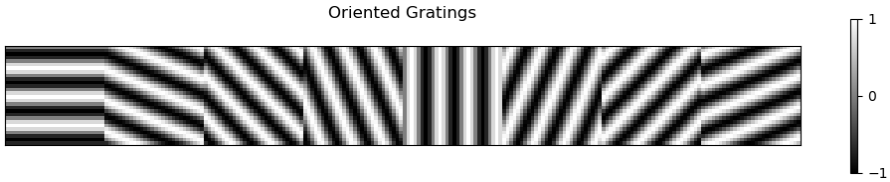
- nonlinear "ot achieve nontrivial results" https://pmc.ncbi.nlm.nih.gov/articles/PMC9141587/
- positive for biological interpretation https://pmc.ncbi.nlm.nih.gov/articles/PMC9141587/
- function should be non symmetrical: https://www.researchgate.net/publication/2308452_Effect_of_Binocular_Cortical_Misalignment_on_Ocular_Dominance_and_Orientation_Selectivity

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)

# Implementation & Problems

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).



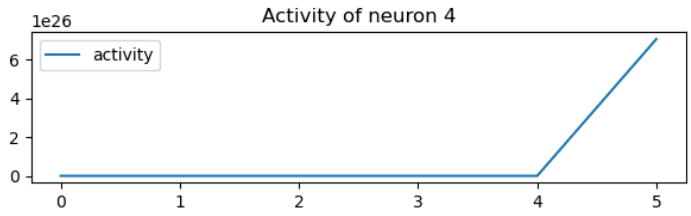

8 orientations evenly spaced over 180°.

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

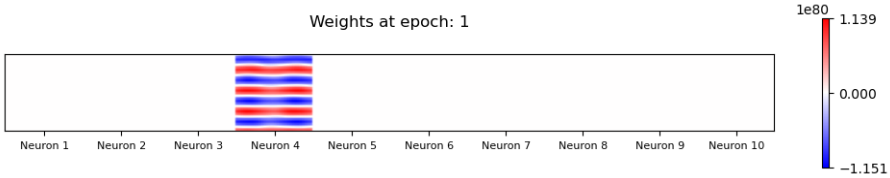
## Problem: Runaway activation

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:



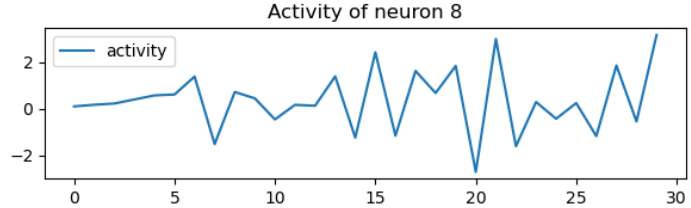

The activity of a single neuron while training.




The weights of all neuros after training.

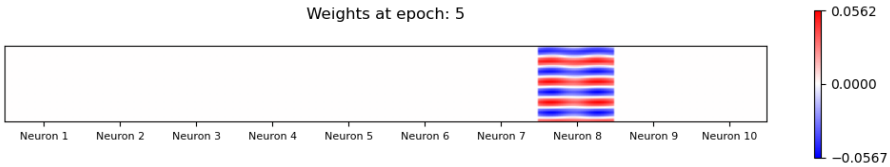
**Solution 1: Weight normalization**

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




The activity of a single neuron while training.

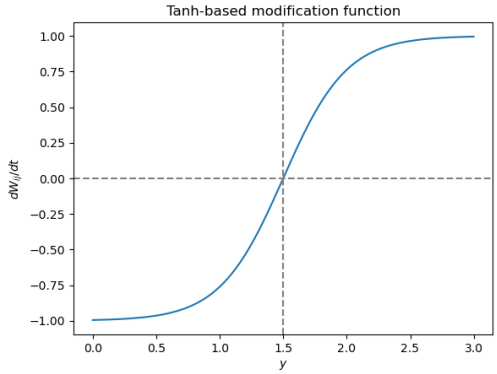
While that does help with the runaway activation problem, we still have the "snowball" problem of one neuron dominating:



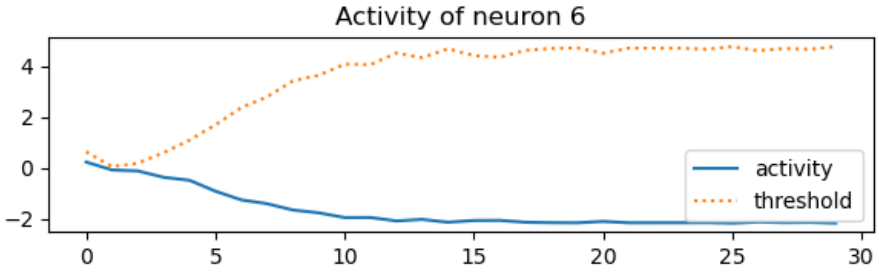

The weights of all neuros after training.

**Solution 2:** $\phi \rightarrow tanh$

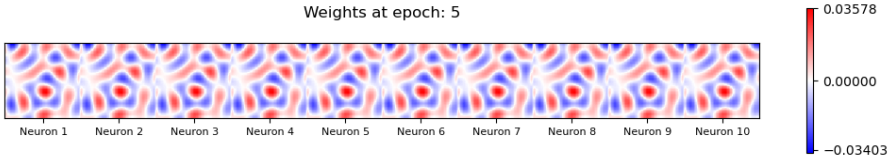
Since some sources portray the modification funtion with a sigmoid-ish trajectory for $y > \theta$, I tried using a $tanh$-based function:




The weights of all neuros after training.




The weights of all neuros after training.

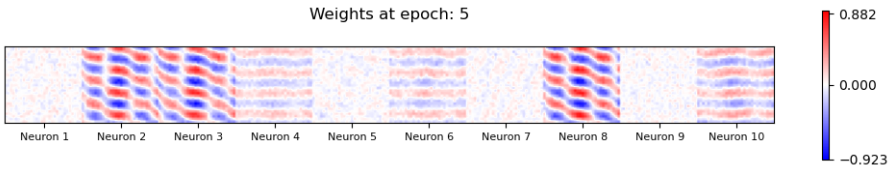



The weights of all neuros after training.

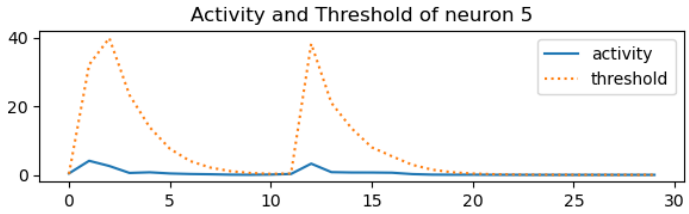
**Solution 3: Curti implementation**

In order to find another solution, I looked at the implemenation for the paper ... that can be found on github ... . The differences to the original BCM imlplementation are:

- adding relu as activation function
- adding moving average of threshold $\theta_t = \gamma\theta_{t-1} + (1-\gamma)\langle z^2 \rangle_{b_t}$
- dividing the weight update by the threshold: $\frac{dw_i}{dt} = \frac{z(z-\theta)x_i}{\theta}$



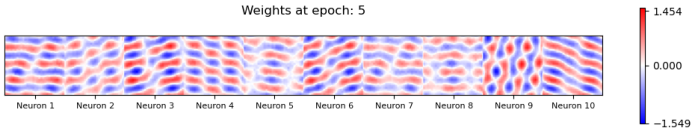

The weights of all neuros after training.



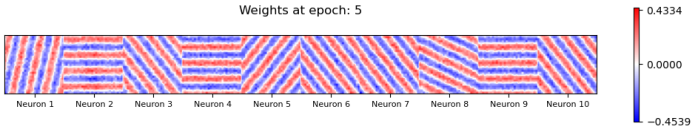

The activity and threshold of a single neuron while training.

## Timescales

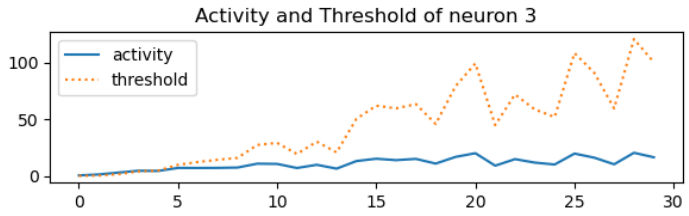
To add another layer of complexity: timescales




for tau_th = 0.1 tau_w = 0.05




for tau_th = 0.1 tau_w = 0.001



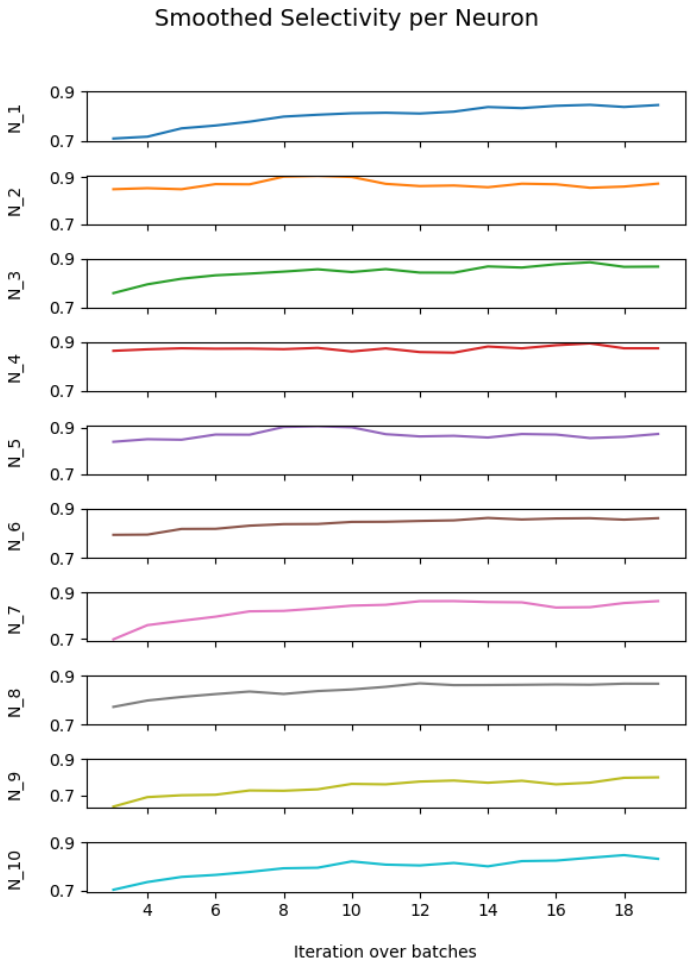

The activity and threshold of a single neuron while training.

The self regulation seems to work in some ways. At t=8 we can see that the spike in activation is picked up by the threshold and that the activation in the next step is reduced. However we can also see that the trend of activity is growing.

# Results and Analysis

Now that

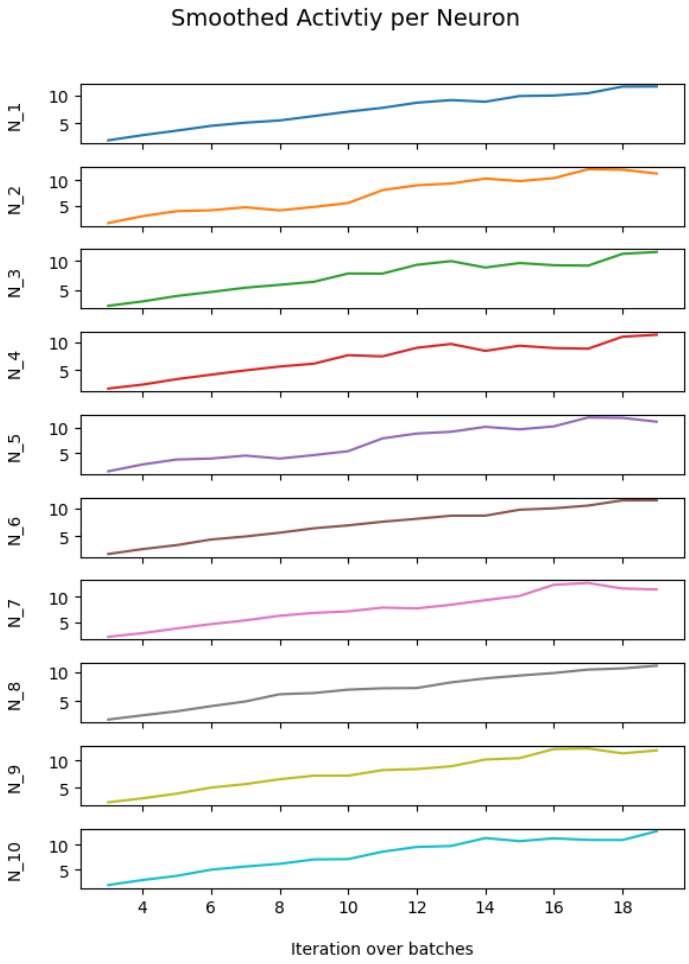
## Selectivity increases with learning




The activity and threshold of a single neuron while training.

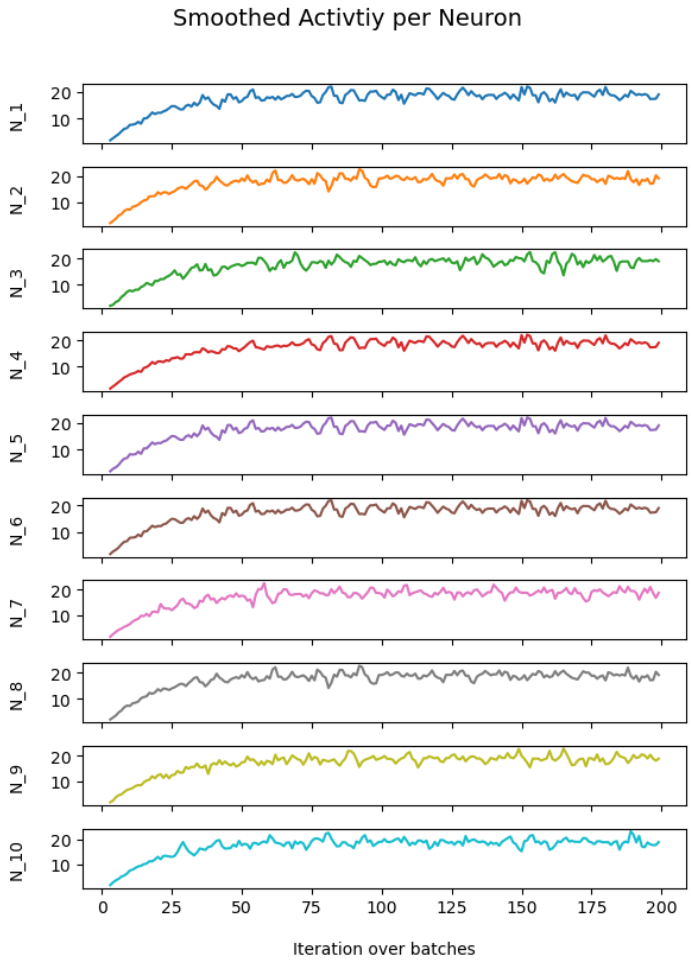
## Stability

ran for 5 times 100 batch size




The activity and threshold of a single neuron while training.

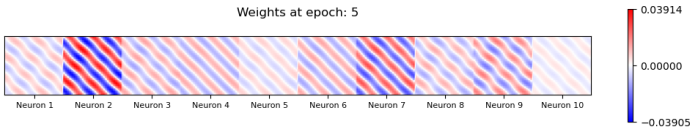
ran for 50 times 100 batch size



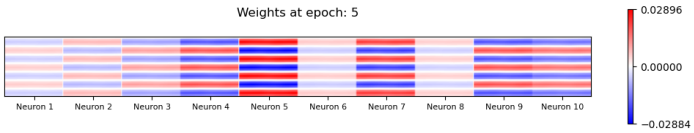

The activity and threshold of a single neuron while training.

# Comparison

## Hebb with weight normalization

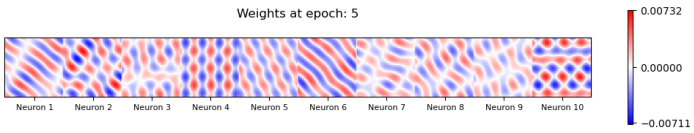



The activity and threshold of a single neuron while training.



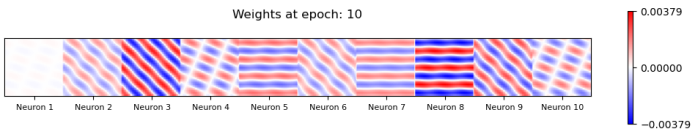

The activity and threshold of a single neuron while training.

## Oja

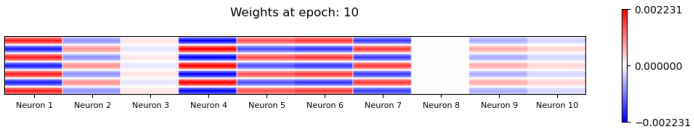



The activity and threshold of a single neuron while training.

letting it run for longer:




The activity and threshold of a single neuron while training.




The activity and threshold of a single neuron while training.

## Future ideas

- WTA & BCM
- timescales

- 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/

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

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