Hebbian neurons, cell assemblies and catastrophic interference

Project report for CS376C: Computational Models of the Neocortex

Abhipray Sahoo
abhipray@stanford.edu
June 16, 2020

1 Introduction

Donald Hebb in his book, "The Organization of Behavior: A Neuropsychological Theory." describes two postulates that aim to describe learning in the brain (Hebb, 1949). These two postulates have to do with synaptic plasticity and the idea of cell assemblies. The first postulate is a description of how synaptic weight strengths change with different inputs.

> "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."

The second postulate describes the idea of cell assemblies which are groups of neurons that fire together when an input stimulus is presented. The activations of these neurons serve to represent the input stimulus.

In the artificial neural networks literature, these two postulates have been explored as learning mechanisms. This paper will describe several weight update rules inspired by the first postulate. Specifically, the Hebbian-LMS algorithm (Widrow et al., 2015) described will be studied in detail. We will also study the second postulate on cell assemblies in the context of ameliorating the problem of catastrophic interference. Catastrophic interference, in the life-long learning literature, is the problem where weights of a network trained on one task decay when trained on another task. This leads to a loss in performance in the first task.

2 Learning rules from Hebb's first postulate

For clarity, we assume that our neuron model performs a weighted sum of various pre-synaptic inputs coming in to produce a post-synaptic output with an activation function.

$$y = \phi(w^T x) \tag{1}$$

The inputs are the elements of the vector x and the weights are summarized by the vector w. ϕ is some activation function; in the simplest form it is identity. For the rest of the discussion, we assume ϕ is identity.

2.1 Activity product rule

Hebb's rule says that when a pre-synaptic neuron fires at the same time as the post-synaptic neuron, the synaptic weight increases. The modification to the weight is time-dependent and local to each synapse. This can be represented mathematically in many forms; the simplest is the activity product rule:

$$\Delta w_i = \mu y x_i \tag{2}$$

The ith synaptic weight changes according to the product of the post-synaptic output y and the ith input x_i. μ is the learning rate. This form of Hebb's rule is local to a synapse and is correlational. However, the weights can grow without bound. Anytime y and x_i are activated at the same time, the weight w_i increases a fixed amount, causing the same input to produce an even larger output y; the weight thus grows exponentially as the output increases.

2.2 Normalized activity product rule

One solution to the unbounded weight growth is to normalize the weight vector after each update.

$$\Delta w_i = \mu y x_i \tag{3}$$
$$w_i(t+1) = w_i(t) + \Delta w_i \tag{4}$$
$$w_i(t+1) = \frac{w_i(t+1)}{||w_i(t)||_2} \tag{5}$$

This ensures that the L2 norm of the weight vector is always 1. The weight vector serves to encode only a direction.

2.3 Weight decay

Another solution to the unbounded weight vector is to introduce a forgetting factor α that causes a weight decay. There are three simple variants:

$$\Delta w_i = \mu y x_i - \alpha w_i \tag{6}$$
$$\Delta w_i = \mu y x_i - \alpha w_i x_i \tag{7}$$
$$\Delta w_i = \mu y x_i - \alpha w_i y \tag{8}$$

Learning rule (6) causes a weight decay independent of the input statistics whereas rules (7) and (8) depend on input statistics. The introduction of weight decay introduces stable points. For example, in (6), we can see that there is a stable point where $w_i = \frac{\mu}{\alpha} y x_i$.

2.4 Covariance rule

A third solution is to use a co-variance product rule:

$$\Delta w_i = \mu(y - E[y])(x_i - E[x_i]) \tag{9}$$

E is the expectation operation. In practice, it can be calculated as a running mean over the past n time-steps. We can see that this rule introduces a stable point when we observe the equation of the line for Δw_i against y. The y-intercept is below 0 at $-\mu(x - E[x])E[y]$. Δw_i can take on both positive and negative values with a stable point where the line meets the x-axis.

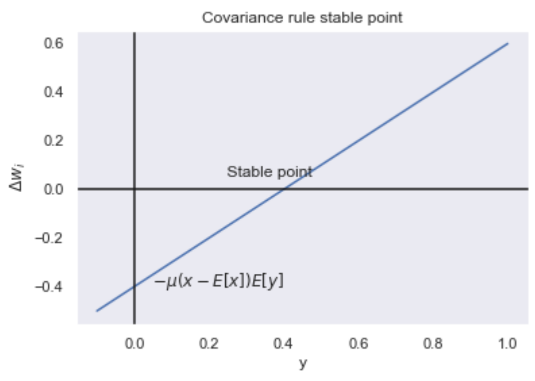

Figure 1: Covariance rule stable point

2.5 Principal and minor components

The common thread across the learning rules so far has been the product of the input vector x and the output y. This can be shown on average to be the correlation between the input and output:

$$\Delta w_i = \mu y x_i \tag{10}$$
$$\Delta w_i = \mu w^T x x_i \tag{11}$$
$$\Delta w_i = \mu \sum_j w_j x_j x_i \tag{12}$$
$$E[\Delta w_i] = \mu \sum_j w_j R_{ij} \tag{13}$$
$$\tag{14}$$

Here R is the correlation matrix $E[xx^T]$. R_{ij} contains the correlation $E[x_i x_j]$. Thus the change in weights is proportional to the correlation matrix R.

For some given data x, the correlation matrix is a second order statistic. Its eigen-decomposition allows us to see which directions have the largest variance. An eigenvector v and eigenvalue λ pair satisfy $Av = \lambda v$. The eigenvectors of the correlation matrix with the largest eigenvalues are called the principal components; the eigenvectors with the smallest eigenvalues are called minor components.

As discussed in (Fyfe, 2007), Miller and Mckay proved the significance of the weight decay terms in terms of eigenvectors of the correlation matrix. They showed that if the weight update rule is of a multiplicative form, then the weight vector converges to a stable point and this stable point is a multiple of the principal eigenvector.

The multiplicative form leading to principal components is

$$\Delta w = Rw - \gamma(w)w \tag{15}$$

Additionally, it was also shown in (Fyfe, 2007) that the anti-hebbian version of the multiplicative rule leads to the weight vector converging to the minor component of the data.

The multiplicative form leading to minor components is

$$\Delta w = -(Rw - \gamma(w)w) \tag{16}$$

2.6 Oja's neuron

Oja introduced a learning rule that aims to find the principal component of the input data:

$$\Delta w_i = \mu y x_i - \mu w_i y^2 \tag{17}$$
$$\tag{18}$$

This is of the multiplicative form of (15). The y^2 term serves to normalize the weights $||w|| = 1$ like the normalized hebbian rule (5).

2.7 Hebbian-LMS

(Widrow et al., 2015) proposed an unsupervised neuron learning rule that was derived from extensions to Hebb's rule. The learning rule is

$$e = \phi(y) - \gamma y \tag{19}$$
$$\Delta w_i = \mu e x \tag{20}$$

where $\phi(y)$ is a sigmoid function like hyperbolic tangent, tanh. This learning rule introduces two stable points where $e = 0$ i.e $\phi(y) = \gamma y$. γ is a constant that determines where the stable equilibrium points are numerically.

We can expand the learning rule equation:

$$\Delta w_i = \mu(\phi(y) - \gamma y)x \tag{21}$$
$$= \mu\phi(y)x - \mu\gamma y x \tag{22}$$

Hebbian-LMS thus takes the form of (16). It leads to the neuron learning the minor component, the direction in the input space explaining the least variance in the data. It can be modified with a sign-reversal to find the principal component.

It is interesting that the Hebbian-LMS neuron derived from extended Hebbian postulates learns the minor component. It fires strongest with an input in the same direction of the minor component. In other words, it detects when a rare input vector shows up. The actual output of the neuron is described as y passing through a half-sigmoid activation.

The paper discusses clustering behavior of the Hebbian-LMS. Once the network converges to the direction of the minor component, it creates a halfspace with the inward normal vector w i.e $\{x|w^T x > 0\}$. Points in this halfspace are the only ones that cause the neuron to fire. Depending on the initial conditions, the weight vector could be pointing in either direction along the minor component. Thus, the input space gets partitioned and the separating boundary is a vector orthogonal to the weight vector.

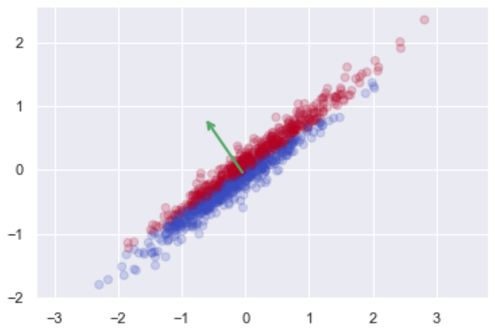

Figure 2: Green arrow is the converged weight vector in the direction of minor component. Red and blue points show the two clusters.

3 Simulations

We simulated the behavior of several Hebbian learning rules described in the previous section. In every case, the weight vector converges to the principal component of the data except with Hebbian-LMS where it converges to the minor component. Each rule used a different learning rate that led to convergence.

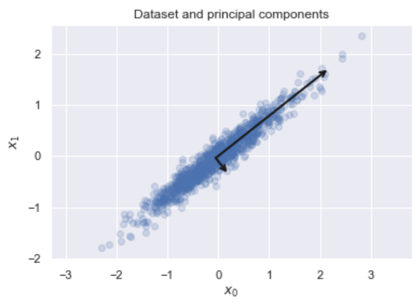

Figure 3: Data along with principal components shown by black arrows

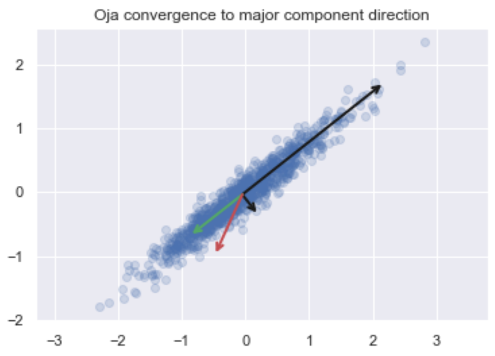

Figure 4: Red arrow is the initial weight vector. Green arrow is the weight vector after convergence with Oja's rule.

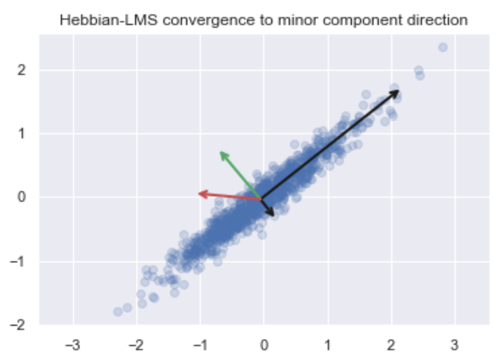

Figure 5: Red arrow is the initial weight vector. Green arrow is the weight vector after convergence with Hebbian-LMS rule.

4 Neural networks

Networks of neurons that follow the above learning rule can be created. In order to create a multi-layer network, the neurons discussed previously will have to be modified, else they end up learning the same principal/minor component direction for their weight vector. This can be done by introducing asymmetry in the weight decay. For example, Oja's rule has been expanded to allow different neurons in the same layer to converge to different principal components. This is called Oja's weighted subspace algorithm:

Different neurons in a single layer have the same inputs but produce different outputs

$$y_i = \sum w^T x$$

$$\Delta w_{ij} = \mu y_i (x_j - \theta_i \sum_{k=1}^{N} y_k w_{kj}) \qquad (23)$$

The weighting factors θ_i can be set to different values with the smallest value leading to the corresponding neuron weight vector converging to the principal component. The learned network effectively projects the input vector into the principal component space.

Such a network required lateral information flow– each output neuron in a layer shared its output with every other neuron in the layer.

Through simulations of networks of Hebbian-LMS, it was shown by (Widrow et al., 2015) that multi-layered networks can learn a good partitioning/clustering of the input space. From our discussion so far, it appears that each neuron's weight vector converges to be parallel to the minor component. Each layer only needs two neurons to encode the two different half-spaces; more neurons would add redundancies. If the first layer had two neurons with opposite weight directions after convergence, inputs to second layer neurons would cluster up along two arms as in figure 6. When a neuron in the second layers learns the minor component of this hidden data distribution, it will cluster together points representing higher firing rates of the two neurons from the first layer.

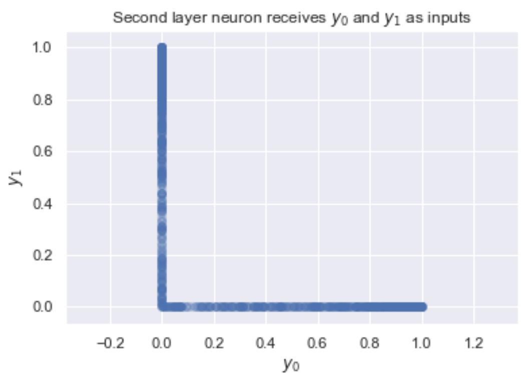

Figure 6: The post-synaptic outputs y_0 and y_1 for two neurons that learn complementary half-spaces. Figure shows the output points for each input point in dataset shown in Figure 3

5 Cell assemblies and catastrophic interference

In the previous section, we looked at how individual neurons might learn using Hebb's first postulate. Hebb's second postulate described cell assemblies as a mechanism for knowledge representation and computation. Cell assemblies can be defined a set of strongly connected neurons. They can be distributed across different brain areas.

(Pulvermüller et al., 2014) referring to cell assemblies as distributed neuronal assembly (DNA) or thought circuit (TC), discuss how they can be carriers of cognition: action, perception, memory, decision making, language etc. Some properties of interest include ignition, reverberation and activity regulation. For a cell assembly to activate, a critical number of neurons need to fire; this causes ignition of most other neurons in the cell assembly. The activity in a cell assembly can reverberate for a period of time and serve to represent it even after the stimulus has ended. Activity regulation is required to keep the network from being fully activated. This can be done with negative feedback in the cortex and sub-cortical regions. Cell assemblies can overlap and serve as mechanisms for relationships between concepts and meanings.

(Pulvermüller et al., 2014) also present a hypothesis for why humans among all species posses vast cognitive abilities. The hypothesis is that cell assemblies form such as to minimize the overlap with other cell assemblies. It does so not by pruning the already existing overlap but by building stronger connections with cross-modal cell assemblies. For example, two cell assemblies representing a similar concept such as ”wish” and ”desire” can have great overlap; to reduce this overlap, the cell assemblies representing the two concepts can grow to include neurons of cell assemblies representing the words ”wish” and ”desire”.

Overlap reduction can help tackle the problem of catastrophic interference. If a cell assembly is used for a task A, and the brain is now learning a new task B using another cell assembly, by reducing the overlap between cell assemblies, it can reduce the possibility of losing ability to perform task A. Additionally, the learning mechanism has to selectively target a cell assembly with a high learning rate and suppress learning rate for cell assemblies near the targeted assembly.

The question arises, could networks of neurons with local Hebbian learning rules self-organize and form cell assemblies? And how can they reduce interference?

The work done by (Tetzlaff et al., 2015) presents two constraints that learning rules have to satisfy in order to allow for the growth of cell assemblies. These are cohesion and competition.

Cohesion is the constraint that if a cell assembly already exists for a given stimulus, i.e has greater weights going into a cell assembly, then the weights inside that cell assembly have to be greater than other cell assemblies also receiving the stimulus. The learning rule has to push the weight vector in the direction where a strong stimulus would cause strengthening of weights between other neurons in the cell assembly. Mathematically, this results in the stable fixed point weight vector being proportional to the product of the input and output. This constraint is not met for all the learning rules we have seen earlier. For example, the Hebbian-LMS rule leads to a fixed point where

$$\sigma(y) = y \tag{24}$$

$$w^* \propto \frac{\sigma(y)}{x} \tag{25}$$

Competition is aimed at reducing overlap with other cell assemblies. If one strong presynaptic stimulus x_i causes postsynaptic activation, its strength increases as per Hebb's rule but it also causes weights for other pre-synpatic inputs x_j to decrease. The fixed point of the rule must be proportional to the inverse of the post-synaptic output y.

$$w^* \propto \frac{1}{y}$$

.

Combining the two constraints leads to the fixed point:

$$w^* = \left(\frac{\mu x y}{\alpha(x - F)}\right)^{\alpha} \tag{26}$$

μ weights the cohesion term and α weights the competition term. α governs the gradient of the fixed point activity function. F is a constant, the homeostatic firing rate. (Tetzlaff et al., 2015) use the following learning rule which leads to the fixed point of (26):

$$\Delta w_i = \mu x_i y + \alpha(F - y)(w_i)^2 \tag{27}$$

They call the first term synaptic plasticity and the second term synaptic scaling. We see immediately, that this can be written in the multiplicative constraint of (15).

$$\Delta w = \mu x y + \alpha(F - y)w^2 \tag{28}$$

$$= \mu x y + (\alpha(F - w^T x)w)w \tag{29}$$

$$\gamma(w) = (\alpha(F - w^T x)w) \tag{30}$$

Thus, like the previous rules we have seen, the Tetzlaff rule leads to the weight vector converging to the direction of the principal component of the input correlation matrix. Unlike the other rules, it meets the additional constraints of cohesion while staying competitive. The synaptic scaling terms prevents different cell assemblies from interfering with each other. They showed how this learning rule can be successfully applied to the formation of cell assemblies that perform complex non-linear computations for a robotic arm tasked with picking and placing an object.

6 Conclusion

We set out to understand how Hebb's learning mechanisms from his two postulates can be used in artificial neural networks. We focused on neuron learning rules that are local to each synapse. Most rules with a weight decay term learn the direction of eigenvectors of the input correlation matrix. Using simulations, we demonstrated this behavior with several rules on a synthetic two-dimensional dataset. We discussed how neurons can be combined to form networks, for example Oja's weighted subspace network or the Hebbian-LMS network. Networks can have sub-networks called cell-assemblies which are sets of strongly connected neurons. Cell assemblies are a powerful construct because they can be grown in a way such as to minimize overlaps and hence ameliorate the problem of catastrophic interference. We then summarized the constraints to local learning rules that lead to the emergence of cell assemblies. Future work includes application of cell assembly growth mechanisms to solve multi-task learning problems.

7 Acknowledgements

Thank you to Prof. Tom Dean for guiding me and for the inception of ideas. I would also like to thank Dr. Widrow for introducing me to his research on applying Hebb's ideas to biological systems.

—

References

Colin Fyfe. 2007. *Hebbian learning and negative feedback networks*. Springer Science & Business Media.

Donald Olding Hebb. 1949. *The organization of behavior: a neuropsychological theory*. J. Wiley; Chapman & Hall.

Friedemann Pulvermüller, Max Garagnani, and Thomas Wennekers. 2014. Thinking in circuits: toward neurobiological explanation in cognitive neuroscience. *Biological cybernetics*, 108(5):573–593.

Christian Tetzlaff, Sakyasingha Dasgupta, Tomas Kulvicius, and Florentin Wörgötter. 2015. The use of hebbian cell assemblies for nonlinear computation. *Scientific reports*, 5:12866.

Bernard Widrow, Youngsik Kim, and Dookun Park. 2015. The hebbian-lms learning algorithm. *ieee ComputatioNal iNtelligeNCe magaziNe*, 10(4):37–53.