# CS 199: Hebbian Plasticity Energy-Based Models

**Lucas Tecot**
Computer Science
UCLA
704611946
lucasmtecot@ucla.edu

## Abstract

We believe that studying and understanding the emergent dynamics of learning methods, and experimenting with how different rules affect emergence, is critical to furthering the field of machine learning. As such, we explore Hebbian plasticity energy-based learning rules, which have been suggested as a biologically-plausible method of deep learning. Specifically we propose methods for how this optimization method can be applied to reinforcement learning environments. This serves as a stepping stone to future work around questioning if these learning rules can create desirable emergent learing behaviors.

## 1 Introduction

At the core of modern neural networks is the concept of emergence. Small computational units bound together and altered by sets of rules and exposed to data create complex behaviors that exhibit traits larger than the sum of their parts. However, there is still a large set of intelligent behaviors that neural networks tend to have issues with, including but not limited to causality, memory, and hierarchical modeling. We desire to more deeply study the rules of these learning algorithms to understand why they cause certain learning properties to emerge, and how we can alter the rules to create new behaviors.

In pursuit of this larger goal, we explore a different form of network optimization, which we broadly categorize as Hebbian plasticity energy-based methods. Specifically we analyze this method in comparison to stochastic gradient descent in order to help understand it's differences in a reinforcement learning environment.

## 2 Motivation

The original motivation behind this line of investigation of came from looking into issues in work surrounding the OpenLock environment [1]. The recent effort of colleagues around this environment involves a causal learner. Essentially this learner generates a data structure of all possible causal chains, and then tests these chains through experience. However, the major problem with this method is it creates an exponential explosion in memory usage. All methods we were considering may have achieved a linear improvement, but nothing that would make this algorithm feasible in the long run.

This led to us considering recurrent networks. In theory, these networks can be thought as a form of memory that is reward-driven. In order to achieve good performance on certain tasks, they need to encode history. Essentially they are translating a non-Markovian environment into a Markovian one, which is similar to how the causal learner works. And this is observed when recurrent networks are trained on simple games [9]. However, if there is a state they can encode that does not allow for improvement, then the network will essentially never "learn" to encode this state, as it's irrelevant in terms of utility. And as such, we felt that working in investigating the fixed point properties of

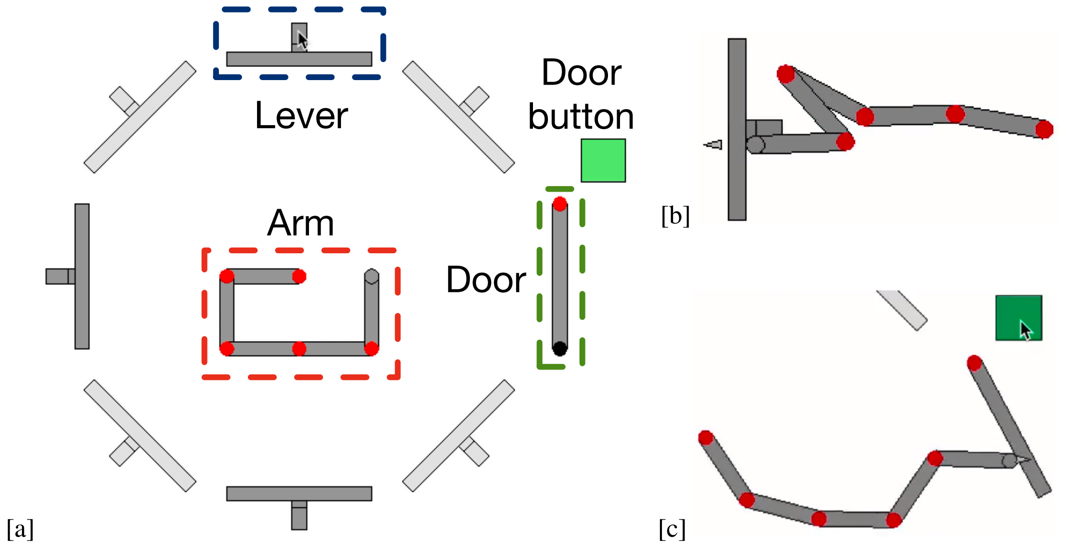


Figure 1: (a) Starting configuration of an OpenLock 3-lever room environment. All levers begin pulled towards the robot arm, whose base is anchored to the center of the display. The arm interacts with levers by either *pushing* outward or *pulling* inward. This is achieved by clicking either the outer or inner regions of the levers' radial tracks, respectively. Only push actions are needed to unlock the door in each lock situation. Light gray levers are always locked, which is unknown to both human subjects and RL at the beginning of training. Once the door is unlocked, the green button can be clicked to command the arm to push the door open. The black circle located opposite the door's red hinge represents the door lock indicator: present if locked, absent if unlocked. (b) Push to open a lever. (c) Open the door by clicking the green button.

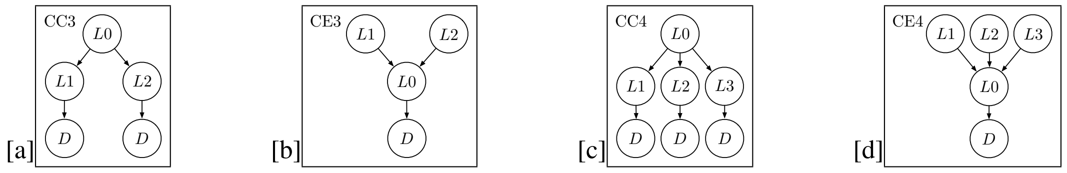


Figure 2: Common cause (CC) and common effect (CE) structures used in OpenLock. $D$ indicates the effect of opening the door. (a) CC3 condition, three lock cues; (b) CE3 condition, three lock cues. (c) CC4 condition, four lock cues; (d) CE4 condition, four lock cues.

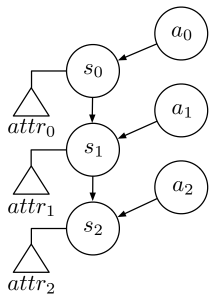


Figure 3: Atomic causal chain. Each $a_i$ represents an action node that can be intervened upon by the agent. Each $s_i$ represents a latent state node which is influenced by attributes, denoted $attr_i$.

recurrent networks could be beneficial for potentially alleviating issues with exponential runoff in the causal learner.

However, we also began to see some work revolving around biologically plausible learning, most of it coming from a concept of Hebbian learning. We became interested in this work from a causal and hierarchical world model learning perspective. Because the update rule is contrastive, and directly either encourages or decourages computational units to have low or high values together, we theorized that maybe this could create a form of heirarchical structure in a model via linked chains of unit clusters. As such, we began work on investigating these methods. The following report outlines the majority of the baseline work to have these methods function in the context of an agent in an environment. However, future work will touch on the topics we discuss here.

## 3 Previous Work

Stochastic gradient descent, although very successful in machine learning, has been criticized for not being a biologically plausible learning optimizer [7]. As such, there are a number of publications that try to bridge this gap and produce machine learning methods closer to the general understanding of biological intelligence. For instance, there are a string of publications that describe a method called equilibrium propagation [5] [2]. This method is conceptually based around an energy based method, in which the goal is to decrease the energy of the states that produce "correct" results, and increase the energy of states that produce "incorrect" results. It does this through altering the weights of artificial neurons that "fire" together, which is the core concept behind the neuroscience theory of Hebbian plasticity. This method is biologically plausible, and the authors provide mathematical proofs for it's equivalence to stochastic gradient descent in terms of optimization [8] [6].

Furthermore, there has been work along the concept of Hebbian plasticity. Miconi et. al. train a classical neural network, but instead add a Hebbian value as an additional training parameter [3]. In this case, the Hebbian value is a simple deterministic running average of the correlation between each pair of artificial neurons. According to their results, this extension of their model allowed for major improvements in one-shot short term memory tasks.

## 4 Method

Scellier and Bengio define the energy function used in equilibrium propagation as follows: [5]

$$E(u) := \frac{1}{2}\sum_i u_i^2 - \frac{1}{2}W_{ij}\rho(u_i)\rho(u_j) - \sum_i b_i\rho(u_i) \tag{1}$$

Where $u$ is the state of the artificial neurons, $W$ are the weights of the network, $b$ are the biases, and $\rho$ is a non-linear activation. Also note that here the weights are symmetrical. Additionally they define a cost function as

$$C := \frac{1}{2}||y - y^{'}||^2 \tag{2}$$

Here we define $u = \{x, h, y\}$, and this cost function is the squared error between the state of the output artificial neurons and the ground truth labels, $y^{'}$. Lastly, they define the total energy as

$$F := E + \beta C \tag{3}$$

Where $\beta$ is the parameter constraining the outputs to the ground truth labels. The network operates by minimizing this energy function for a specific input, output ground truth, and $\beta$. As such, the network can be optimized to have low error by minimizing the energy of states that produce a low cost, and maximizing the energy of states that produce a high cost. Scellier and Bengio achieve this through a derivative of the following loss function with respect to the network parameters.

$$L := -\frac{1}{\beta}(F^{\beta} - F^{0}) \tag{4}$$

Where $F^{\beta}$ is the total energy minimized with $\beta = \beta$, and $F^0$ is the total energy minimized with $\beta = 0$. This produces the following weight update, which is reminiscent of a Hebbian update rule.

$$\Delta W_{i,j} \propto \lim_{\beta \to 0} \frac{1}{\beta}(\rho(u_i^{\beta})\rho(u_j^{\beta}) - \rho(u_i^0)\rho(u_j^0)) \tag{5}$$

In practice, this update is achieved through a two-phase process. First the network's energy is minimized with respect to $u$ and with the outputs unconstrained. Then the process is repeated with constrained outputs. Finally, this update rule is applied this with respect to the network parameters.

In this paper, we alter this formulation to operate in the context of reward for reinforcement learning environments. We attempt three different forms of algorithms.

## 4.1 DQN Equilibrium Propagation

This method exactly follows the deep Q-network introduced by Mnih et. al. [4]. Their method uses a single network to estimate the value of taking an action in a specific state. Our method operates on the exact same principles, except the underlying network used to estimate values of state-action pairs is optimized through the equilibrium propagation method described above, as opposed to standard stochastic gradient descent on a cost function.

## 4.2 Policy Equilibrium Propagation

Similarly, this method follows the REINFORCE algorithm introduced by Sutton et. al. [10], which utilities one critic model that estimates the value of a state, and one policy model that selects which action to take given a state. In our method, the critic model is implemented as a classical neural network. However, we optimize the policy network using equilibrium propagation. The loss typically used in the policy network is

$$L = \sum_t -\log \pi_{\theta}(a_t|s_t)v_t \tag{6}$$

Where $\pi$ is the policy model and $v_t$ is the advantage. The advantage is calculated using the temporal difference rule of $v_t = [r_t + \gamma V(s_{t+1})] - V(s_t)$, where $V$ is the critic network, $\gamma$ is a future reward discount parameter, and $r_t$ is the reward observed at time $t$. The intuition here is that we wish to select actions which lead to rewards that are higher than the average of the state we are in. Any higher than average rewards will produce a positive advantage, which in turn incentives a higher probability of the taken action. Likewise, a lower than average reward will produce a negative advantage, which incentives a lower probability of the action.

Similarly, in equilibrium propagation, we wish to optimize the network parameters such that states that tend to produce a high advantage have a low energy, and states that produce a low advantage have a high energy. As such, we define our loss function as follows:

$$L = \sum_t E_t \log \pi_{\theta}(a_t|s_t)v_t \tag{7}$$

Note that in this equation, we optimize through taking the derivative with respect to the parameters in the energy function, and treat the log-likelyhood as a constant. As described before, we can also implement this through the equilibrium propagation update rule by recording the correlations between the units in the network at each time-step, multiplying these correlations by the log-likelyhood and advantage, and adding this to the weights.

## 4.3 Reward Equilibrium Propagation

Lastly, we introduce a method that relies solely on the observed reward, and does not rely on a critic network. Similarly to Miconi et. al., we can store a running sum of all the parameter updates at every time-step [3]. If we update our parameters according to this running sum for the reward we observe at every time-step, the cumulative effect will be an update proportional to the true observed value of each state. Furthermore, in order to prevent a run-off optimization where weights increase to infinity and energy is globally minimized, we also subtract the running average of all previous updates in the network. As such, the net effect is a decrease in energy of states the perform better with respect to the global average. We formulate the updates as follows.

$$\Delta\theta \propto \sum_t r_t\bar{\theta}_t - \hat{r_t}\hat{\theta}_t \tag{8}$$

$$\bar{\theta}_t = \eta\bar{\theta}_{t-1} + \Delta\theta_t^{'} \tag{9}$$

$$\hat{\theta}_t = \gamma\hat{\theta}_{t-1} + (1-\gamma)\Delta\theta_t^{'} \tag{10}$$

$$\hat{r}_t = \gamma\hat{r}_{t-1} + (1-\gamma)r_t \tag{11}$$

Were $\eta, \gamma \in [0, 1]$. We define $\Delta\theta_t^{'}$ as the negative derivative of the energy function with respect to the network parameters at time $t$, which can be expressed as

$$\Delta W_{i,j} \propto \rho(u_i)\rho(u_j) \tag{12}$$

$$\Delta b_i \propto \rho(u_i) \tag{13}$$

## 5 Evaluation

We tested each of these methods against the OpenAI Gym Cartpole environment, in which your model must control a cart such that it keeps a pole placed on the cart upright for as long as possible. Although future work should and will test against much more complex and interesting environments, in this work Cartpole is used merely as a tool to measure the baseline viability of each method as a learning algorithm.

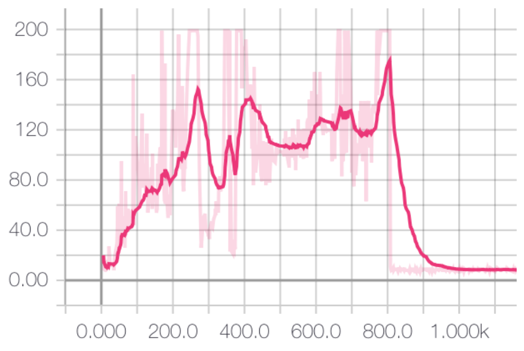
### 5.1 Implementation Details

Every model used in this work, whether or not it is optimized through stochastic gradient descent on loss or equilibrium propagation, is implemented using a multi-layer perceptron with a single hidden layer.

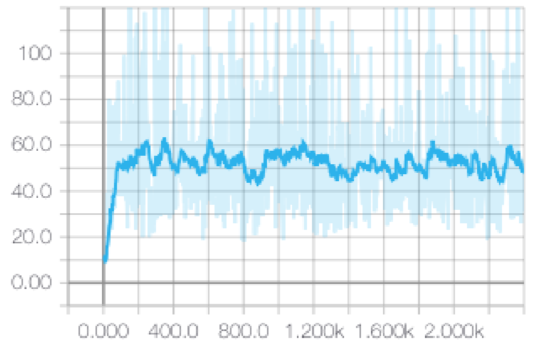
In equilibrium propagation, when we desire to produce an output we optimize the model's units towards an energy minimum. In the case of DQN equilibrium propagation, this is done through gradient descent on the energy function. However, in policy and reward equilibrium propagation, where the outputs are never constrained but the inputs are always constrained, we simply feed forward the network values as we would with a normal neural network, as this will produce the energy minimum.

Similarly, when we optimize the network parameters in all the equilibrium propagation techniques, we can either manually update the weights and biases using the rules described in the reward equilibrium propagation section, or we can use an optimizer that performs gradient descent on some form of combination of energies from various states in the learning process. Although functionally equivalent, in practice, for the value based method we were able to achieve the described results using an optimizer. In the reward based method, the acheived results are through the manual update rules. Furthermore, in the reward based method, we found it more beneficial to negate the update on the biases. This means that we increase the biases according to the running average activations, and decrease them according to the rewards observed during each episode. Although we theorize that this could be tied to other neuroscience concepts such as long term potentiation and long term depression, ultimately at this time this is an implementation detail which we cannot fully explain analytically.

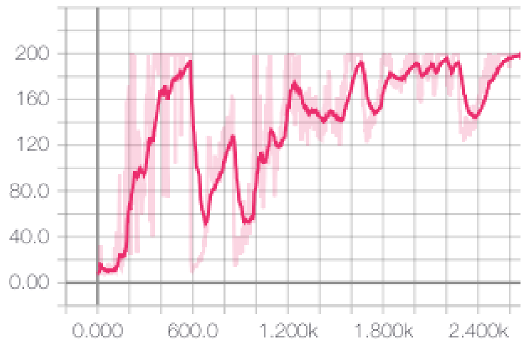
The link to the code and additional details can be found in the appendix.



(a) Policy Equilibrium Propagation



(b) Reward Equilibrium Propagation



(c) Actor-Critic Baseline

Figure 4: Plots of total episode reward (vertical axis) with respect to number of environment episodes (horizontal axis) in the Cartpole environment. Reward is equivalent to number of time steps the pole is balanced. Note that the environment is considered solved at 200 reward. Although not shown here, the Reward Equilibrium Propagation method does slowly decline in performance.

## 5.2 Results

As is demonstrated in Figure 4, the performance of the reward equilibrium propagation isn't comparable in this example to a baseline actor-critic method. It seems to plateau at a small reward quickly and not optimize from there. However, policy equilibrium propagation does manage to solve the environment with a similar training curve to the baseline. It does have strongly divergent behavior, as can be seen from the sharp cutoffs in performance during training, but this demonstrates promise for it's ability as a general learning algorithm.

There are a few interesting aspects we observed during training that we wish to address. Firstly, all methods have cutoffs or declines in performance at a certain point in training. This tends to occur when suddenly the network has a sharp increase in the probability of a specific action being taken, irregardless of the input state. We also observed high sensitivity to hyper-parameters in training. Although these features aren't necessarily uncommon in general neural networks, equilibrium propagation seems to have very fragile stability.

Secondly, the DQN equilibrium propagation method had similar convergence issues. In the case of equilibrium propagation, we were not able to produce a model that achieve performance above a random choice policy. We observed that the cost of the model begins to converge but then dramatically shoots up at a certain stage in training. We suspect this may be due to changing ratios between the combined energy and cost as training progresses, but ultimately we did not do more detailed analysis in this work.

## 6 Future Work

Although the proposed methods are interesting due to some measure of biological plausibility, they will need to be iterated on to produce practical learn-ability. Future works must address this and provide results across a variety of environments. However, there are a number of other interesting areas that we wish to see addressed in future works.

Firstly, in this paper, the majority of tests were run using a standard feed-forward network. However, we believe that interesting behaviors may emerge in networks that have recurrent connections. As demonstrated by Miconi et. al. [3], Hebbian learning rules may allow improvements in memory tasks. Specifically we'd like to test against environments that strongly benefit from short-term memory [1], and run fixed point analysis on the network.

Secondly, there may be ways to approximate the energy minimization process and provide speedups. Currently these methods require two gradient optimizations: one for minimizing energy, and one for updating the parameters. Although this is relatively manageable with a feed-forward network, it will become increasingly computationally intensive with more complexly connected networks. Finding rules to approximate the process could improve performance. Additionally, the learning rule for these methods don't require the back-propagation of gradients, and rely only on correlation values. This could allow for improved speed in optimizing extremely large networks.

Lastly, we believe that Hebbian-based optimization techniques may encourage more hierarchical relations within the networks themselves. Because units that fired together are either encouraged or discouraged to fire together again through updates in weights, it's possible we will converge on model parameters that cause high levels of segmentation between units for different tasks. Although in itself this isn't necessarily an improvement, observing this behavior could be an avenue for future work on hierarchical modeling and catastrophic forgetting. As such, we would like to see future work that analyzes these method's tendency to form hierarchy within their networks.

## 7 Conclusion

In the interest of understanding more deeply the emergent learning properties of learning algorithms, we have proposed methods and experimented with biologically plausible forms of optimization. Although they require additional work to provide practical utility, it's our hope that continuations of this work will gleam deeper insights into our current methods, and lead to further hypothesis on how to deal with their shortcomings.

## References

[1] Mark Edmonds, James Kubricht, Colin Summers, Yixin Zhu, Brandon Rothrock, Song-Chun Zhu, and Hongjing Lu. Human causal transfer: Challenges for deep reinforcement learning.

[2] T. Mesnard, W. Gerstner, and J. Brea. Towards deep learning with spiking neurons in energy based models with contrastive Hebbian plasticity. *ArXiv e-prints*, December 2016.

[3] Thomas Miconi, Jeff Clune, and Kenneth O. Stanley. Differentiable plasticity: training plastic neural networks with backpropagation. *CoRR*, abs/1804.02464, 2018.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[5] B. Scellier and Y. Bengio. Equilibrium Propagation: Bridging the Gap Between Energy-Based Models and Backpropagation. *ArXiv e-prints*, February 2016.

[6] B. Scellier, A. Goyal, J. Binas, T. Mesnard, and Y. Bengio. Generalization of Equilibrium Propagation to Vector Field Dynamics. *ArXiv e-prints*, August 2018.

[7] Benjamin Scellier and Yoshua Bengio. Towards a biologically plausible backprop. *CoRR*, abs/1602.05179, 2016.

[8] Benjamin Scellier and Yoshua Bengio. Equivalence of equilibrium propagation and recurrent backpropagation. *CoRR*, abs/1711.08416, 2017.

[9] David Sussillo and Omri Barak. Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, 25(3):626–649, 2013. PMID: 23272922.

[10] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

## 8 Appendix

```
Code: https://github.com/ltecot/emergence_properties
Commit used to generate graphs in this paper:
c1fca92a807a941c25ba42655d16db1ed214ed3e
Additional literature review: https://github.com/ltecot/papers
```