Learning by Competition of Self-Interested Reinforcement Learning Agents

Stephen Chung

Department of Computer Science, University of Massachusetts Amherst, USA
minghaychung@umass.edu

Abstract

An artificial neural network can be trained by uniformly broadcasting a reward signal to units that implement a REINFORCE learning rule. Though this presents a biologically plausible alternative to backpropagation in training a network, the high variance associated with it renders it impractical to train deep networks. The high variance arises from the inefficient structural credit assignment since a single reward signal is used to evaluate the collective action of all units. To facilitate structural credit assignment, we propose replacing the reward signal to hidden units with the change in the L^2 norm of the unit's outgoing weight. As such, each hidden unit in the network is trying to maximize the norm of its outgoing weight instead of the global reward, and thus we call this learning method *Weight Maximization*. We prove that Weight Maximization is approximately following the gradient of rewards in expectation. In contrast to backpropagation, Weight Maximization can be used to train both continuous-valued and discrete-valued units. Moreover, Weight Maximization solves several major issues of backpropagation relating to biological plausibility. Our experiments show that a network trained with Weight Maximization can learn significantly faster than REINFORCE and slightly slower than backpropagation. Weight Maximization illustrates an example of cooperative behavior automatically arising from a population of self-interested agents in a competitive game without any central coordination.

Introduction

The *error backpropagation algorithm* (backprop) efficiently computes the gradient of an objective function with respect to parameters by iterating backward from the last layer of a multi-layer artificial neural network (ANN). However, backprop is generally regarded as being biologically implausible (Crick 1989; Mazzoni, Andersen, and Jordan 1991; O'Reilly 1996; Bengio et al. 2015; Hassabis et al. 2017; Lillicrap et al. 2020). First, the learning rule given by backprop is non-local, as it relies on information other than input and output of a neuron-like unit computed in the feedforward phase. Second, backprop requires synaptic symmetry in the forward and backward paths, which has not been observed in biological systems. Third, backprop requires precise coordination between the feedforward and feedback phase because the feedforward value has to be retained until the error signal arrives.

Alternatively, REINFORCE (Williams 1992), a special case of $A_{R-\lambda P}$ when $\lambda = 0$ (Barto and Anandan 1985), could be applied to all units as a more biologically plausible way of training a network. It is shown that the learning rule gives an unbiased estimate of the gradient of return (Williams 1992). Another interpretation of this relates to viewing each unit as a reinforcement learning (RL) agent, with each agent trying to maximize the same reward from the environment. We can thus view an ANN as a *team of agents* playing a cooperative game, a scenario where all agents receive the same reward. Such a team of agents is also known as *coagent network* (Thomas 2011). However, coagent networks can only feasibly solve simple tasks due to the high variance associated with this training method and thus the low speed of learning. The high variance stems from the lack of structural credit assignment, i.e. a single reward signal is used to evaluate the collective action of all agents.

To address the lack of structural credit assignment in a team of agents trained by REINFORCE, we consider delivering a different reward signal to each hidden unit instead of the same global reward. Here hidden units refer to the units that output to other units in the network instead of to the environment. As such, each hidden unit is associated with a vector of weights by which the unit's actions influence other units in the network, and we call this vector the *outgoing weight*. We propose to replace the global reward signal to each hidden unit with the change in the L^2 norm of its outgoing weight, such that each hidden unit in the network is trying to maximize the norm of its outgoing weight. We call this new learning method *Weight Maximization*. This is based on the intuition that the norm of a unit's outgoing weight roughly reflects the contribution of the unit in the network. This change of reward signals turns the original cooperative game into a competitive game since units no longer receive the same reward.

We prove that Weight Maximization is approximately following the gradient of return in expectation, showing that every hidden unit maximizing the norm of its outgoing weight also approximately maximizes the network's rewards. This illustrates an example of cooperative behavior automatically arising from a population of self-interested agents in a competitive game and offers an alternative perspective of training an ANN—each unit maximizing the norm of its outgoing weight instead of a network maximizing its performance. This alternative perspective localizes the optimization problem for each unit, yielding a wide range of RL solutions in training a network. Our experiments show that a network trained with Weight Maximization can learn much faster than REINFORCE, such that its speed is slightly lower than backprop.

One may question whether the change of synaptic strength, analogous to the change of weights in ANNs, can be used to guide plasticity in biological systems, since the change of synaptic strength has a much slower timescale compared to the activation of neurons. To address this issue, we generalize Weight Maximization to use eligibility traces, such that the network can still learn when the outgoing weights change slowly. Weight Maximization with eligibility traces also solves the three aforementioned problems of backprop regarding biological plausibility. Nonetheless, the biological plausibility of Weight Maximization remains to be investigated. It is not yet clear if there exists a molecular mechanism that uses the change of synaptic strength in axons to guide the change of synaptic strength in dendrites.

In summary, our paper has the following main contributions:

- We propose a novel algorithm called Weight Maximization that allows efficient structural credit assignment and significantly lowers the variance associated with REINFORCE when training a team of agents.
- We prove that Weight Maximization is approximately following the gradient of return in expectation, establishing the approximate equivalence of hidden units maximizing the norm of their outgoing weights and external rewards, thus providing theoretical justification for algorithms (Uhr and Vossler 1961; Klopf and Gose 1969; Anderson 1986; Selfridge 1988) based on the norm of outgoing weights.
- Weight Maximization can be used to train both continuous-valued and discrete-valued units, offering an advantage over backprop.
- Weight Maximization represents a feasible alternative to backprop given their comparable learning speed.
- We generalize Weight Maximization to use eligibility traces, which solves several major issues of backprop relating to biological plausibility.

The paper and the appendix are available at https://arxiv.org/abs/2010.09770. The code is available at https://github.com/stephen-chung-mh/weight_max.

Notation

We consider a Markov Decision Process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma, d_0)$, where $\mathcal{S}$ is a finite set of states of an agent's environment (although this work can be extended to the infinite state case), $\mathcal{A}$ is a finite set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition function giving the dynamics of the environment, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, $\gamma \in [0, 1]$ is a discount factor, and $d_0 : \mathcal{S} \to [0, 1]$ is an initial state distribution. Denoting the state, action, and reward signal at time t by S_t, A_t, and R_t respectively, $P(s, a, s') = \Pr(S_{t+1} = s'|S_t = s, A_t = a)$, $R(s, a) = \mathbb{E}[R_t|S_t = s, A_t = a]$, and $d_0(s) = \Pr(S_0 = s)$, where P and d_0 are probability mass functions. An episode is a sequence of states, actions, and rewards, starting from $t = 0$ and continuing until reaching a terminal state. For any learning method, we can measure its performance as it improves over multiple episodes, which makes up a run.

Letting $G_t = \sum_{k=t}^{\infty} \gamma^{k-t} R_k$ denote the infinite-horizon discounted return accrued after acting at time t, we are interested in finding, or approximating, a *policy* $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ such that for any time $t > 0$, selecting actions according to $\pi(s, a) = \Pr(A_t = a|S_t = s)$ maximizes the expected return $\mathbb{E}[G_t]$. The value function for policy π is V^π where for all $s \in \mathcal{S}$, $V^\pi(s) = \mathbb{E}[G_t|S_t = s, \pi]$, which can be shown to be independent of t.

In this paper, we restrict attention to policies computed by a network of L stochastic units. The following definitions hold for any time $t > 0$. Let $H_t^{(l)}$ denote the activation value of the unit $l \in \{1, 2, ..., L\}$ at time t, which is a one-dimensional discrete or continuous random variable. We also let $H_t^{(0)} = S_t$ and $A_t = H_t^{(L)}$. We call unit l, where $1 \leq l \leq L - 1$, a hidden unit and unit L the output unit. For any $1 \leq l \leq L$, the distribution of $H_t^{(l)}$ conditional on $H_t^{(0:l-1)}$ is given by $\pi_l : (\mathcal{S} \times \mathbb{R}^{l-1}) \times \mathbb{R} \to [0, 1]$, such that $\Pr(H_t^{(l)} = h^{(l)}|H_t^{(0:l-1)} = h^{(0:l-1)}; \mathbf{w}^{(l)}) = \pi_l(h^{(0:l-1)}, h^{(l)}; \mathbf{w}^{(l)})$, where $\mathbf{w}^{(l)}$ is the parameter of unit l. To sample an action A_t from the network given S_t, we iteratively sample $H_t^{(l)} \sim \pi_l(H_t^{(0:l-1)}, \cdot; \mathbf{w}^{(l)})$ from $l = 1$ to L. In other words, the network is a feedforward network with units ordered and connections restricted to higher-numbered units. This formulation is a generalization of multi-layer networks of stochastic units since multi-layer networks are the special case of units being arranged in layers and connections between non-adjacent layers being set to zero. We mostly focus on multi-layer networks in this paper.

We assume that for all $1 \leq l \leq L$, $\pi_l(h^{(0:l-1)}, h^{(l)}; \mathbf{w}^{(l)})$ can be expressed as a differentiable function $f_l(\mathbf{w}^{(l)} \cdot h^{(0:l-1)}, h^{(l)})$, where $\cdot$ denotes the dot product, and call the vector $\mathbf{w}^{(l)}$ the *incoming weight* of unit l. We denote the weight connecting from unit k to l (where $k < l$) as $\mathbf{w}_{(k)}^{(l)}$, and call the vector $\mathbf{v}^{(k)} = [\mathbf{w}_{(k)}^{(k+1)}, \mathbf{w}_{(k)}^{(k+2)}, ..., \mathbf{w}_{(k)}^{(L)}]^T$ the *outgoing weight* of unit k.

We denote $||\mathbf{x}||_p^p$ as the p-norm of vector $\mathbf{x}$ to the power of p, and $x^{(m:n)}$ as $\{x^m, x^{m+1}, ..., x^n\}$.

Algorithm

The gradient of return at time t with respect to $\mathbf{w}^{(l)}$, where $1 \leq l \leq L$, can be estimated by REINFORCE, also known as the likelihood ratio estimator:

$$\nabla_{\mathbf{w}^{(l)}} \mathbb{E}[G_t] = \sum_{k=t}^{\infty} \gamma^{(k-t)} \mathbb{E}[G_k \nabla_{\mathbf{w}^{(l)}} \log \Pr(A_k|S_k)]. \quad (1)$$

We can show that each term of the summation also equals the expectation of the update given by REINFORCE applied to a hidden unit with the same reinforcement signal G_t:

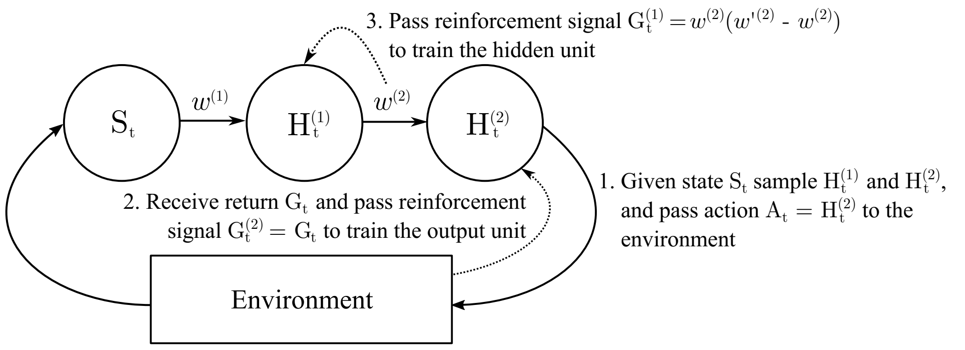
3. Pass reinforcement signal $G_t^{(1)} = w^{(2)}(w'^{(2)} - w^{(2)})$ to train the hidden unit
S_t
$w^{(1)}$
$H_t^{(1)}$
$w^{(2)}$
$H_t^{(2)}$
1. Given state S_t sample $H_t^{(1)}$ and $H_t^{(2)}$, and pass action $A_t = H_t^{(2)}$ to the environment
2. Receive return G_t and pass reinforcement signal $G_t^{(2)} = G_t$ to train the output unit
Environment

Figure 1: Illustration of a network with two units trained by Weight Maximization. See text for explanation.

Theorem 1. *Let the policy be a network of stochastic units as defined above. For all $t > 0$ and $1 \le l \le L$,*

$$\mathbb{E}[G_t \nabla_{\mathbf{w}^{(l)}} \log Pr(A_t|S_t)] = \mathbb{E}[G_t \nabla_{\mathbf{w}^{(l)}} \log \pi_l(H_t^{(0:l-1)}, H_t^{(l)}; \mathbf{w}^{(l)})]. \quad (2)$$

See Williams (1992) for the proof. This shows that we can apply REINFORCE to each unit of the network, and the learning rule still gives an unbiased estimate of the gradient of the return. Therefore, denoting α as the step size, we can update parameters by the following stochastic gradient ascent rule:

$$\mathbf{w}^{(l)} \leftarrow \mathbf{w}^{(l)} + \alpha G_t \nabla_{\mathbf{w}^{(l)}} \log \pi_l(H_t^{(0:l-1)}, H_t^{(l)}; \mathbf{w}^{(l)}). \quad (3)$$

However, this learning rule suffers from high variance since a single reinforcement signal (G_t) is used to evaluate the collective action of all units. In other words, the signal G_t has a weak correlation with the gradient $\nabla_{\mathbf{w}^{(l)}} \log \pi_l(H_t^{(0:l-1)}, H_t^{(l)}; \mathbf{w}^{(l)})$, making the multiplication of these two terms have a high variance. To reduce this variance, we propose to replace this signal to each unit l by:

$$G_t^{(l)} = \begin{cases} \mathbf{v}^{(l)} \cdot \Delta \mathbf{v}_t^{(l)} & \text{for } l \in \{1, 2, ..., L-1\}, \\ G_t & \text{for } l = L. \end{cases} \quad (4)$$

where $\cdot$ denotes the dot product, and $\Delta \mathbf{v}_t^{(l)}$ is the change of the outgoing weight, $\mathbf{v}^{(l)}$, resulting from the update of the outgoing weight at time t. For the output unit, we let $G_t^{(L)} = G_t$; that is, the output unit is still maximizing the return from the environment.

With the new reinforcement signal $G_t^{(l)}$, each hidden unit is approximately maximizing the L^2 norm of its outgoing weight. To see this, consider the change in the L^2 norm of the outgoing weight for hidden unit l at time t:

$$||\mathbf{v}^{(l)} + \Delta \mathbf{v}_t^{(l)}||_2^2 - ||\mathbf{v}^{(l)}||_2^2 \quad (5)$$
$$= 2\mathbf{v}^{(l)} \cdot \Delta \mathbf{v}_t^{(l)} + ||\Delta \mathbf{v}_t^{(l)}||_2^2. \quad (6)$$

We observe that $G_t^{(l)}$ is proportional to (6) except for the term $||\Delta \mathbf{v}_t^{(l)}||_2^2$. We choose to ignore this term in the reinforcement signal since this term is $\mathcal{O}(\alpha^2)$ while $2\mathbf{v}^{(l)} \cdot \Delta \mathbf{v}_t^{(l)} = \mathcal{O}(\alpha)$. If the step size α is very small as in typical experiments, then $||\Delta \mathbf{v}_t^{(l)}||_2^2$ is also negligible and does not affect experimental results. However, by adjusting α, this term can be made arbitrarily small or large, and so we choose to remove it completely instead. This removal is necessary to arrive Theorem 2.

The motivation for using the change in the norm of a unit's outgoing weight as a reinforcement signal in (4) is based on the idea that the norm of a unit's outgoing weight roughly reflects the contribution of the unit in the network. For example, if the hidden unit's output is useful in guiding action, then the output unit will learn a large weight associated with it. Conversely, if the hidden unit is outputting random noise, then the output unit will learn a zero weight associated with it. With the new reinforcement signal, each unit gets a different local reward that evaluates its own action instead of the entire team's action, thus allowing efficient structural credit assignment. This idea of measuring the worth of a unit by its outgoing weight's norm has a long history as we discuss in the Related Work section below.

With the new reinforcement signal, the learning rule at time t becomes:

$$\mathbf{w}^{(l)} \leftarrow \mathbf{w}^{(l)} + \Delta \mathbf{w}_t^{(l)}, \quad (7)$$
$$\Delta \mathbf{w}_t^{(l)} = \alpha G_t^{(l)} \nabla_{\mathbf{w}^{(l)}} \log \pi_l(H_t^{(0:l-1)}, H_t^{(l)}; \mathbf{w}^{(l)}), \quad (8)$$

where $1 \le l \le L$. Note that the only difference of the above learning rule with (3) is the change of the reinforcement signals to hidden units. We call this new learning method *Weight Maximization*. To apply the learning rule, we compute $\Delta \mathbf{w}_t^{(l)}$ iteratively from $l = L$ to 1. The pseudo-code can be found in Algorithm 1 of Appendix B. The computational cost of Weight Maximization is the same as backpropagation since it is linear in the number of layers.

Though we restrict our attention to a network with a single output unit in our formulation, Weight Maximization can be generalized to a network with multiple output units easily by only replacing the rewards to all hidden units by the change in the norm of their outgoing weights in (4).

To illustrate the algorithm, we consider an example of a simple network with only two units (one hidden unit and one output unit), as shown in Figure 1. For every time step t, the algorithm performs the following steps:

1. Given state S_t, we sample the activation value of hidden unit $H_t^{(1)} \sim \pi_1(S_t, \cdot; w^{(1)})$. For example, if the unit is a Bernoulli-logistic unit and the state S_t is one-dimensional, then $\Pr(H_t^{(1)} = 1|S_t) = \sigma(w^{(1)} S_t)$ and $\Pr(H_t^{(1)} = 0|S_t) = 1 - \sigma(w^{(1)} S_t)$, where σ is the sigmoid function. We sample output unit $H_t^{(2)} \sim \pi_2(H_t^{(1)}, \cdot; w^{(2)})$ similarly, and pass action $A_t = H_t^{(2)}$ to the environment.
2. After receiving return G_t from the environment (which is only known at the end of an episode, but can be replaced with TD error for online learning), we use it to train the output unit by REINFORCE: $w'^{(2)} = w^{(2)} + \alpha G_t^{(2)} \nabla_{w^{(2)}} \log \pi_2(H_t^{(1)}, H_t^{(2)}; w^{(2)})$, where α is the step size and $G_t^{(2)} = G_t$.
3. We compute the reinforcement signal to the hidden unit by $G_t^{(1)} = v^{(1)} \Delta v^{(1)} = w^{(2)}(w'^{(2)} - w^{(2)})$, which is used to train the hidden unit by REINFORCE: $w'^{(1)} = w^{(1)} + \alpha G_t^{(1)} \nabla_{w^{(1)}} \log \pi_1(S_t, H_t^{(1)}; w^{(1)})$.

In the following, we discuss the theoretical properties of Weight Maximization.

Goal Alignment Condition

In this section, we address an important question: Under which situations are the goals of the hidden units aligned with the goal of the whole network? To simplify the discussion, we only consider single-time-step MDPs in this section and drop the subscript t, but the theorems here can be generalized to multiple-time-step MDPs.

To answer this question, we analyze the gradient followed by the learning rule of the hidden units. First, the output unit is maximizing the return G as a result of Theorem 1 and REINFORCE:

$$\mathbb{E}[\Delta \mathbf{w}^{(L)}] \propto \nabla_{\mathbf{w}^{(L)}} \mathbb{E}[G]. \tag{9}$$

Then consider the learning rule of unit $L-1$, which is maximizing $G^{(L-1)}$:

$$\begin{aligned} \mathbb{E}[\Delta \mathbf{w}^{(L-1)}] &\propto \nabla_{\mathbf{w}^{(L-1)}} \mathbb{E}[G^{(L-1)}] & (10)\\ &= \nabla_{\mathbf{w}^{(L-1)}} \mathbb{E}[\mathbf{v}^{(L-1)} \cdot \Delta \mathbf{v}^{(L-1)} \cdot] & (11)\\ &\propto \nabla_{\mathbf{w}^{(L-1)}} (\mathbf{v}^{(L-1)} \cdot \nabla_{\mathbf{v}^{(L-1)}} \mathbb{E}[G]). & (12) \end{aligned}$$

The last line is due to the fact that $\Delta \mathbf{v}^{(L-1)}$ is an entry in the vector $\Delta \mathbf{w}^{(L)}$, and so we can substitute the expectation of it with (9). We can continue the same process to derive the formulas of $\mathbb{E}[\Delta \mathbf{w}^{(l)}]$ for $l = L-2, L-3, ..., 1$. This shows that the learning rule of hidden units is related to high-order cross partial derivatives of $\mathbb{E}[G]$ instead of the first-order derivative. To have the goal of units and the network aligned, it is sufficient and necessary that the cross partial derivatives are the same as the first-order derivative. Formally,

Lemma 1. *Let the policy be a network of stochastic units as defined above, and $\Delta \mathbf{w}^{(l)}$ be defined by (8). Then $\mathbb{E}[\Delta \mathbf{w}^{(l)}] \propto \nabla_{\mathbf{w}^{(l)}} \mathbb{E}[G]$ for all $1 \leq l \leq L-1$ if and only if $\nabla_{\mathbf{w}^{(l)}} (\mathbf{v}^{(l)} \cdot \nabla_{\mathbf{v}^{(l)}} \mathbb{E}[G]) \propto \nabla_{\mathbf{w}^{(l)}} \mathbb{E}[G]$ for all $1 \leq l \leq L-1$.*

The proof can be found in Appendix A.1. Therefore, we define the *goal alignment condition*, which is a sufficient condition that the hidden units are also maximizing the global return when applying Weight Maximization, by:

Definition 1. *Let the policy be a network of stochastic units as defined above. We say that the network has satisfied the goal alignment condition in an MDP if for all $1 \leq l \leq L-1$,*

$$\nabla_{\mathbf{w}^{(l)}} (\mathbf{v}^{(l)} \cdot \nabla_{\mathbf{v}^{(l)}} \mathbb{E}[G]) \propto \nabla_{\mathbf{w}^{(l)}} \mathbb{E}[G].$$

Except in special cases like a piecewise linear network with a piecewise linear reward function, this goal alignment condition does not hold exactly. However, the goal alignment condition holds approximately for all networks without extra assumptions:

Theorem 2. *Let the policy be a network of stochastic units as defined above. For all $1 \leq l \leq L-1$,*

$$\nabla_{\mathbf{w}^{(l)}} (\mathbf{v}^{(l)} \cdot \nabla_{\mathbf{v}^{(l)}} \mathbb{E}[G]) = \nabla_{\mathbf{w}^{(l)}} \mathbb{E}[G] + \mathcal{O}(||\mathbf{v}^{(l)}||_2^2).$$

The proof can be found in Appendix A.2. Therefore, the learning rule of hidden unit l is only approximately following the gradient of return in expectation with an error of $\sum_{k=l+1}^{L} \mathcal{O}(||\mathbf{w}^{(k)}||_2^2)$, since the error accumulates across units. In other words, the bias associated with the learning rule scales with the L^2 norm of the unit's outgoing weight. It is interesting to see that the more 'successful' a unit is (measured by the norm of its outgoing weight), the more severe is the problem of goal misalignment.

To combat the problem of goal misalignment, we suggest adding L^2 regularization, or weight decay, which can be seen as a soft constraint on the L^2 norm of weights (Goodfellow, Bengio, and Courville 2016) and thus prevents weights from having a large magnitude.

By replacing the rewards to the hidden units with (4) and adding weight regularization, the original cooperative game is turned into a *competitive game*. A competitive game refers to the scenario where each agent is receiving different rewards (Sutton and Barto 2018). With weight regularization, the outgoing units have a 'limited' norm of weight to allocate due to the soft constraint, and so the units connected to the same downstream unit have to compete for the limited resources. In other words, units want to maximize their outgoing weights' norm, but the outgoing units want to minimize their incoming weights' norm, and therefore competition exists between units.

Weight Maximization with Eligibility Traces

Though Weight Maximization does not require a separate feedback pathway, the learning of a hidden unit needs to wait for the outgoing weight to finish updating; in contrast, in biological neurons, the change of synaptic strength has a slower timescale than the activation of neurons (Nicoll 2017), making it difficult to be used as an immediate feedback signal. Besides different timescales, the change in synaptic strength of a biological neuron is the result of the neuron's activity within a time interval instead of a single discrete time step. For example, multiple pairs of spikes are required to induce noticeable change in synaptic weights in spike-timing-dependent plasticity (STDP) experiments (Citri and

	Multiplexer		CartPole		Acrobot		LunarLander	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
Weight Max	0.81	0.01	390.38	43.25	-97.05	2.90	111.23	16.58
Weight Max w/ traces	n.a.	n.a.	373.11	17.86	-105.00	4.38	39.04	14.39
REINFORCE	0.29	0.01	163.92	64.43	-134.15	8.87	-94.23	19.03
REINFORCE with Thomas (2011)	0.28	0.02	363.71	24.07	-112.26	8.85	-66.19	67.14
STE Backprop	0.76	0.01	420.73	18.30	-98.55	5.05	47.53	35.04
Backprop	0.84	0.01	411.82	25.98	-91.82	5.24	71.77	21.02

Table 1: Average return over all episodes.

Malenka 2008; Gerstner et al. 2014). In short, to be closer to biologically-observed plasticity rules, the effect of a unit's action on the outgoing weight should be slow and long-lasting instead of immediate and precise as assumed in Weight Maximization. We propose to use eligibility traces to solve both issues.

In the following discussion, we only consider a multi-layer network of stochastic units consisting of $M \geq 1$ layers. The L units in the network are arranged into a multi-layer structure such that the weights connecting units in non-adjacent layers are frozen to zero. We also denote $d(l)$ as the layer in which unit l resides and assume the last layer contains only the output unit.

Assume each unit requires a single time step to compute the weight update. That is, the weight update of unit l at time t, denoted by $\Delta \mathbf{w}_t^{(l)}$, is based on $G_{t-1}^{(l)}$ (defined in (4)) and $H_{t-1}^{(l)}$, the reward and action at the previous time step. Since $\Delta \mathbf{w}_t^{(l)}$ does not depend on the weight change at the same time step, we can compute it for all layers in parallel. However, the reward for the hidden layer m is lagging behind by $M - m$ time steps, as it takes a single time step for each of the $M - m$ upper layers to compute their weight updates. Therefore, the first issue—the effect of a unit's action on the outgoing weight should be slow—can be seen as the problem of delayed reward; the action of a hidden unit on layer m at time t affects its rewards at time $t + M - m$, but not before.

For the second issue, consider the scenario where the effect of a unit's action is long-lasting on the change of its outgoing weight. For example, if the output unit learns with decaying eligibility traces (Sutton and Barto 2018), then all $\Delta \mathbf{w}_{t+1}^{(L)}, \Delta \mathbf{w}_{t+2}^{(L)}, ...$ depends on the action of units on layer $M-1$ at time t, though the dependence decays with time. In other words, the action of layer $M-1$ at time t will affect the change in its outgoing weight at and after time $t+1$. We can continue the discussion for layer $M-2, M-3, ..., 1$. From this perspective, the second issue—the effect of a unit's action on the outgoing weight should be long-lasting—can again be seen as the problem of delayed reward; the action of a hidden unit on layer m at time t affects all its rewards at and after time $t + M - m$, but not before.

The problem of delayed reward is well studied in RL, and one prominent and one biologically plausible solution is to use eligibility traces. We suggest using the following decay function $\lambda^l(t) : \mathbb{Z} \to [0, 1]$ for the unit l:

$$\lambda^l(t) = \begin{cases} 0 & \text{for } t \leq M - d(l) - 1, \\ (1-\lambda)\lambda^{t-(M-d(l))} & \text{else,} \end{cases} \tag{13}$$

where $\lambda \in [0, 1]$ is the decay rate and $d(l)$ is the layer in which unit l resides. Therefore, $\lambda^l(t)$ is the exponentially decaying trace but shifted by $M - d(l)$ time steps, since the action of unit l does not affect the reward in the next $M-d(l)$ time steps.

With this decay function, we can generalize Weight Maximization to use eligibility traces. The learning rule of *Weight Maximization with eligibility traces* for a multi-layer network of stochastic units at time t is given by :

$$\mathbf{w}^{(l)} \leftarrow \mathbf{w}^{(l)} + \Delta \mathbf{w}_t^{(l)}, \tag{14}$$

$$\Delta \mathbf{w}_t^{(l)} = \alpha \delta_{t-1}^{(l)} \mathbf{z}_{t-1}^{(l)}, \tag{15}$$

$$\delta_t^{(l)} = \begin{cases} \mathbf{v}^{(l)} \cdot \Delta \mathbf{v}_t^{(l)} & \text{for } l \leq L-1, \\ R_t + \gamma V^\pi(S_{t+1}) - V^\pi(S_t) & \text{for } l = L, \end{cases} \tag{16}$$

$$\mathbf{z}_t^{(l)} = \sum_{k=0}^{t-1} \lambda^l(k) \gamma^{k-(M-d(l))} \nabla_{\mathbf{w}^{(l)}} \log \pi_l(H_{t-k}^{(0:l-1)}, H_{t-k}^{(l)}; \mathbf{w}^{(l)}), \tag{17}$$

where α denotes the step size and $1 \leq l \leq L$, $V^\pi(s)$ denotes the state value $\mathbb{E}[G_t | S_t = s, \pi]$. (17) computes the eligibility traces $\mathbf{z}_t^{(l)}$ by summing over the gradients of the log probability of the selected action multiplied by both the decay rate $\lambda^l(k)$ and the discount rate $\gamma^{k-(M-d(l))}$. (16) computes the reinforcement signal $\delta_t^{(l)}$ delivered to the units on layer l, which equals the TD error $R_t + \gamma V^\pi(S_{t+1}) - V^\pi(S_t)$ (we replace G_t in (4) with TD error to make the algorithm online) for the last layer and the change in the norm of outgoing weights for hidden layers. Lastly, the weight update (15) is given by the product between the eligibility traces and the reinforcement signal scaled by the step size α.

In practice, the state values $V^\pi(S_t)$ is generally unknown, but we can estimate the state values by another network implementing a TD algorithm, which is called a critic network (Sutton and Barto 2018). The pseudo-code can be found in Algorithm 2 of Appendix B.

With the above modification, the effect of a unit's action on its outgoing weight is slow and long-lasting, in accordance with biologically-observed plasticity rules. Weight Maximization with eligibility traces also solves the three problems of backprop discussed in the introduction. Its learning rule is local and does not require any separate feedback pathways. The algorithm can be implemented in parallel for all layers, and there are no distinct feedforward and feedback phases for the whole network.

It should be noted that the feedback signal in backprop can also be computed based on the change of a unit's outgoing weight in some cases, eliminating the need for a separate feedback pathway. However, as discussed earlier, it is not biologically plausible to use the change of a unit's outgoing weight as an immediate feedback signal. The solution presented in this section, namely eligibility traces, can only be applied to Weight Maximization but not to backprop, since backprop requires the activation values to be precisely matched with the feedback signals at the same time step. This underlines one major difference between the two algorithms.

Related Work

Research on solving tasks by a team of RL agents has a long history. Tsetlin et al. (1973) described learning automata in team and game problems, followed by stochastic learning automata (Narendra and Thathachar 1974, 2012). Klopf (1972, 1982) proposed the hedonistic neuron hypothesis, which conjectured that individual neurons seek to maximize their own pleasure, and the collective behavior of these neurons can yield powerful adaptive systems. This idea was further developed by Seung (2003) in spiking neural networks. Barto and Anandan (1985), Barto (1985) and Barto and Jordan (1992) introduced the $A_{R-\lambda P}$ algorithm and showed that a team of $A_{R-\lambda P}$ units could learn with a globally-broadcast reward signal. Extending this class of learning rule, Williams (1992) introduced REINFORCE, a special case of $A_{R-\lambda P}$ when $\lambda = 0$, and proved that a team of agents trained with REINFORCE ascends the average reward gradient. Such a team of agents is recently called coagent networks (Thomas 2011). Theories relating to training coagent networks have been investigated (Thomas 2011; Kostas, Nota, and Thomas 2020; Thomas and Barto 2011), and Thomas (2011) proposed a variance reduction method for training coagent networks with REINFORCE by disabling exploration randomly. Chung (2021) proposed the MAP propagation algorithm, which minimizes the energy of the network before applying REINFORCE, to reduce the variance efficiently. However, in these papers, each agent in the team receives the same reward signal. In contrast, here we propose that each agent works to maximize the norm of its outgoing weight instead of a common reward signal, which transforms the problem from a *cooperative game* into a *competitive game*. Zhang, Yang, and Başar (2021) reviewed recent development in the wider field of multi-agent RL.

Measuring the worth of a unit by the norm of its outgoing weight has been proposed (Uhr and Vossler 1961; Klopf and Gose 1969; Selfridge 1988). In these papers, hidden units with small outgoing weights are replaced by new hidden units with random incoming weights. These methods are thus based on an evolutionary approach instead of the RL approach as in Weight Maximization. Anderson (1986) proposed A_{R-P} *algorithm with Penalty Prediction*, which pushes units with small outgoing weights to match the incoming values.

In addition, there is a large literature on methods for training networks of stochastic units, and a review can be found in Weber et al. (2019). STE backprop (Bengio, Léonard, and Courville 2013) is a practical method of training a network of stochastic discrete units. Though STE backprop does not follow the gradient of the loss function, it is arguably the most effective way of training quantized ANN (Courbariaux, Bengio, and David 2015; Rastegari et al. 2016) and Yin et al. (2018) provides some theoretical justification for STE backprop. However, STE backprop suffers the same problem with backprop regarding biological plausibility.

Besides a team of agents trained by REINFORCE, many biologically plausible alternatives to backprop have been proposed. Biologically plausible learning rules based on reward prediction errors and attentional feedback have been proposed (Pozzi, Bohte, and Roelfsema 2020; Roelfsema and Ooyen 2005; Rombouts, Bohte, and Roelfsema 2015); but these learning rules mostly require a non-local feedback signal. Moreover, local learning rules based on contrastive divergence or nudging the values of output units have been proposed (Movellan 1991; Hinton 2002; Scellier and Bengio 2017). See Lillicrap et al. (2020) for a comprehensive review of algorithms that approximate backprop with local learning rules based on the differences in units' values. Contrary to these papers, Weight Maximization does not require any separate feedback pathways or distinct phases in learning. Also, most of these algorithms are applied in supervised or unsupervised learning tasks, while Weight Maximization is applied in both RL tasks and supervised learning tasks[1].

Experiments

We applied our algorithms to four RL tasks: multiplexer, CartPole, Acrobot, and LunarLander. The multiplexer task is a simple toy task with a single time step, where the agent is rewarded $+1$ if it outputs the correct answer and -1 if it outputs the incorrect answer[2]. Details of the tasks can be found in Appendix C. We tested two variants of Weight Maximization: i. Weight Maximization, ii. Weight Maximization with Eligibility Traces. For both variants, we used Algorithm 2 in Appendix B (i. corresponds to the case $\lambda = 0$). We did not test Weight Maximization with Eligibility Traces on the multiplexer task since the task only has a single time step.

All networks considered have the same architecture: a three-layer network of stochastic units, with the first hidden layer having 64 units, the second hidden layer

[1]Any supervised learning tasks can be converted to RL tasks, though it may not be optimal since the knowledge of the loss function is not utilized.

[2]We assume the reward function to be unknown to the agent, so this task is considered an RL task instead of a supervised learning task.

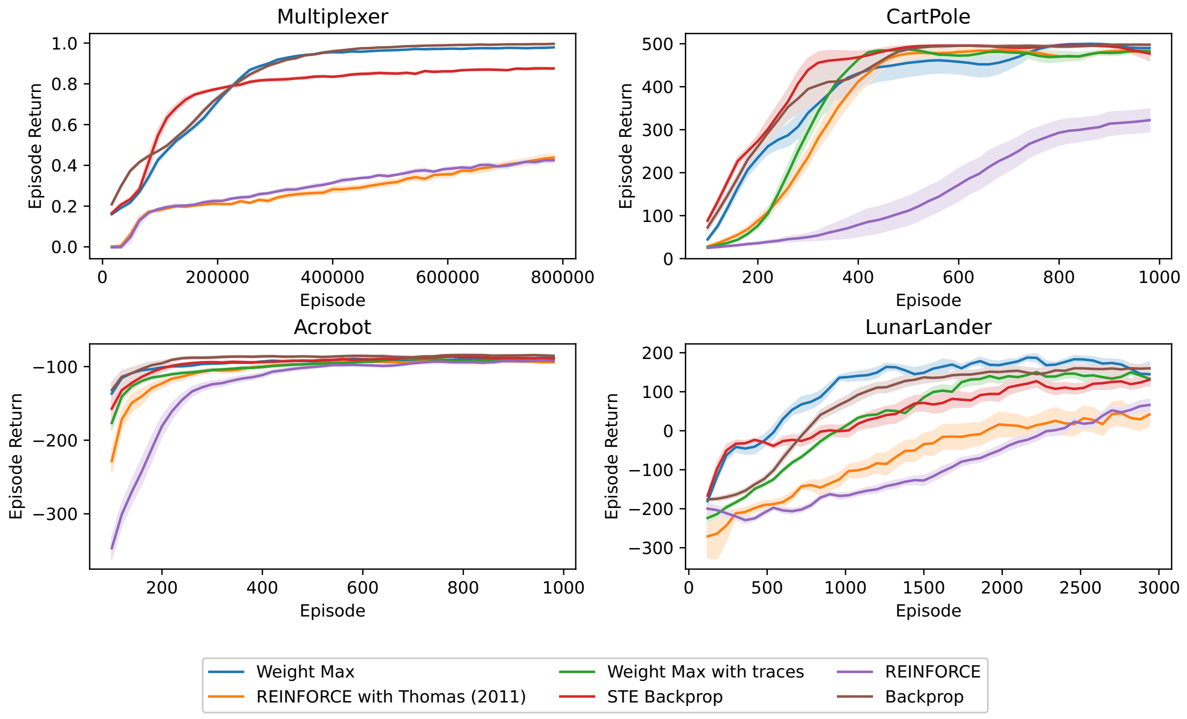

Figure 2: Episode returns in different RL tasks. Results are averaged over 10 independent runs, and shaded areas represent standard deviation over the runs. Curves are smoothed with a running average of 100 episodes (10,000 episodes for the multiplexer task).

having 32 units, and the output layer being a softmax layer. All hidden units are Bernoulli-logistic units, i.e. $\pi_l(h^{(0:l-1)}, h^{(l)}, \mathbf{w}^{(l)}) = \sigma(\mathbf{w}^{(l)} \cdot h^{(0:l-1)})^{h^{(l)}}(1 - \sigma(\mathbf{w}^{(l)} \cdot h^{(0:l-1)}))^{1-h^{(l)}}$, where σ is the sigmoid function.

We consider four baselines to train hidden units: i. REINFORCE, ii. REINFORCE with the variance reduction method proposed by Thomas (2011), iii. STE backprop (Bengio, Léonard, and Courville 2013) and iv. backprop. For i. to iii., the networks are the same as the one used in Weight Maximization. For iv., the Bernoulli-logistic units are replaced with deterministic rectified linear units (ReLUs) so the hidden units can be trained by backprop. In all baselines, the output unit was trained by REINFORCE, and we used eligibility traces. Thus, all learning methods in the experiments are variants of Actor-Critic with Eligibility Traces (episodic) (Sutton and Barto 2018) with different methods of accumulating trace (for the baselines) or different reward signals to each unit (for Weight Maximization). For the critic networks in all experiments, we used a three-layer ANN trained by backprop. Other experiments' details can be found in Appendix C.

The average return over ten independent runs is shown in Fig 2. The mean and standard deviation of the average return can be found in Table 1. We observe that both variants of Weight Maximization have a significantly better performance than REINFORCE, suggesting that Weight Maximization allows effective structural credit assignment. Also, Weight Maximization has a similar performance compared to STE backprop, indicating that it is an effective method for training discrete units.

However, compared to a network of continuous-valued units trained by backprop, Weight Maximization performed slightly worse (except for the LunarLander). This is likely due to the limitation of discrete-valued units—units can communicate only by means of binary values instead of real values. This view is supported by the observation that discrete-valued units trained by STE backprop also performed worse than backprop. However, discrete units have the advantages of low memory and communication costs; thus the slower learning may be compensated by a more efficient computation.

We notice that adding traces to Weight Maximization does not improve the performance. This is likely due to the more difficult credit assignment problem introduced by this approach. By using traces, the change in a unit's outgoing weight at time t is affected by the unit's action at times $t, t-1, t-2, \ldots$ instead of only time t, making temporal credit assignment more difficult and thus learning slower.

In addition, we found that the representation learned by Weight Maximization is more statistically independent than backprop. For example, after training an agent to solve CartPole by Weight Maximization (STE backprop), the absolute correlation across units on the first and the second layer is 0.690 (0.888) and 0.640 (0.830) respectively on average. This may be explained by the dynamics of Weight Maximization—units compete with each other to be more 'useful'

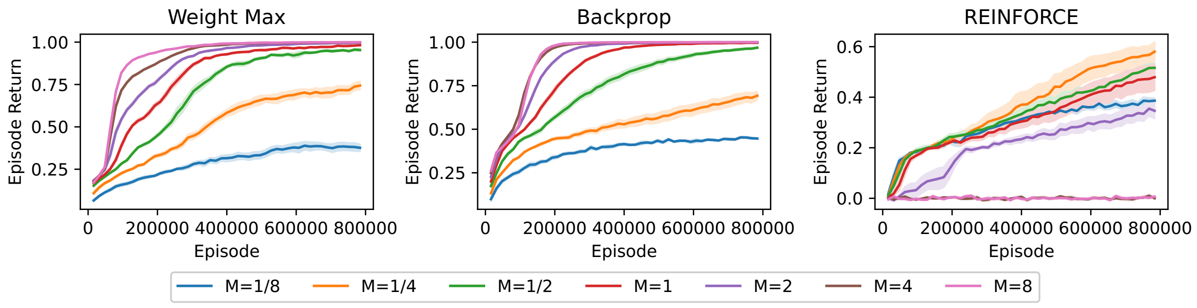

Figure 3: Episode returns in the multiplexer task with a varying number of units in the network. Results are averaged over 10 independent runs, and shaded areas represent standard deviation over the runs. Curves are smoothed with a running average of 10,000 episodes.

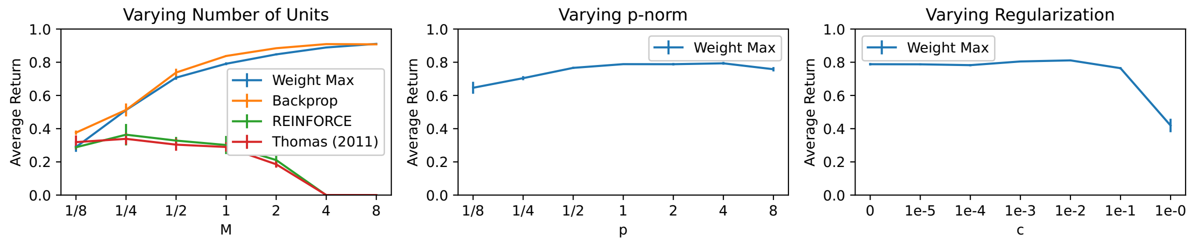

Figure 4: Average returns of all episodes in the multiplexer task with different hyperparameters. Results are averaged over 10 independent runs.

and so have to learn different signals.

To better understand how different learning rules scale with the number of units in a network, we repeated the experiments on the multiplexer task with varying numbers of units in the network. Let the network have $64M$ and $32M$ units in the first and the second layer of the network respectively. The experimental results with different M can be found in Figure 3 and Figure 4.

We observe that the performance of both Weight Maximization and backprop improve with a larger M, while the performance of REINFORCE peaks at $M = 1/4$. This illustrates a critical issue of REINFORCE that renders it impractical to training large networks—as all units are independently exploring and a single reward signal is used to evaluate the collective exploration of all units, it is more difficult to assign credit to each unit with a larger network. Thus, a larger network leads to a higher variance of parameter updates and slower learning. This issue is alleviated in Weight Maximization since each unit receives a different reward signal that is strongly correlated with the unit's own action.

In Weight Maximization we replace the reward to each unit by the change in the L^2 norm of the outgoing weight as shown in (4). It is also possible to use L^p norm instead of L^2 norm by generalizing (4) (see details in Appendix C). The experimental results on the multiplexer tasks with varying p can be found in Figure 4. The results show that the network is also able to learn with a similar performance when using other L^p norms.

We tested the effects of L^2 weight regularization, with $c \geq 0$ being the strength of weight regularization. That is, we add $-2c\mathbf{w}^{(l)}$ to the learning rule (7). The experimental results on the multiplexer tasks with varying c can be found in Figure 4. We observe that the performance peaks at $c = 0.01$, suggesting that a small weight regularization can improve performance. This is in line with our analysis of the goal alignment condition.

Future Work and Conclusion

The approximate equivalence of every unit maximizing the global return and every unit maximizing the norm of its outgoing weight offers a wide range of possible methods to train ANNs besides backprop. Since each hidden unit faces a local optimization problem when trying to maximize the norm of its outgoing weight, learning rules apart from REINFORCE can also be applied to train hidden units. The lack of central coordination in Weight Maximization also leads to the possibility of implementing the algorithm asynchronously and efficiently with neuromorphic circuits (Indiveri et al. 2011).

In conclusion, we propose a novel algorithm that reduces the variance associated with training a team of agents with REINFORCE and thus significantly increases the learning speed. The proposed algorithm solves several major problems of backprop relating to biological plausibility. We also analyze the theoretical properties of the proposed algorithm and establish that training hidden units to maximize their norm of outgoing weights is approximately equivalent to training them to maximize a global external reward signal.

Acknowledgments

We would like to thank Andrew G. Barto, who inspired this research and provided valuable insights and comments.

References

Anderson, C. W. 1986. *Learning and problem solving with multilayer connectionist systems*. Ph.D. thesis, Citeseer.

Barto, A. G. 1985. Learning by statistical cooperation of self-interested neuron-like computing elements. *Human Neurobiology*, 4(4): 229–256.

Barto, A. G.; and Anandan, P. 1985. Pattern-recognizing stochastic learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, (3): 360–375.

Barto, A. G.; and Jordan, M. I. 1992. Gradient following without back-propagation in layered networks. *et-al. Frontiers in cognitive neuroscience*, 443–449.

Bengio, Y.; Lee, D.-H.; Bornschein, J.; Mesnard, T.; and Lin, Z. 2015. Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*.

Bengio, Y.; Léonard, N.; and Courville, A. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Chung, S. 2021. MAP Propagation Algorithm: Faster Learning with a Team of Reinforcement Learning Agents. *Advances in Neural Information Processing Systems*, 34.

Citri, A.; and Malenka, R. C. 2008. Synaptic plasticity: multiple forms, functions, and mechanisms. *Neuropsychopharmacology*, 33(1): 18–41.

Courbariaux, M.; Bengio, Y.; and David, J.-P. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, 3123–3131.

Crick, F. 1989. The recent excitement about neural networks. *Nature*, 337(6203): 129–132.

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

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep learning*. MIT press.

Hassabis, D.; Kumaran, D.; Summerfield, C.; and Botvinick, M. 2017. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2): 245–258.

Hinton, G. E. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8): 1771–1800.

Indiveri, G.; Linares-Barranco, B.; Hamilton, T. J.; Van Schaik, A.; Etienne-Cummings, R.; Delbruck, T.; Liu, S.-C.; Dudek, P.; Häfliger, P.; Renaud, S.; et al. 2011. Neuromorphic silicon neuron circuits. *Frontiers in neuroscience*, 5: 73.

Klopf, A. H. 1972. *Brain function and adaptive systems: a heterostatic theory*. 133. Air Force Cambridge Research Laboratories, Air Force Systems Command, United States Air Force.

Klopf, A. H. 1982. *The hedonistic neuron: a theory of memory, learning, and intelligence*. Toxicology-Sci.

Klopf, A. H.; and Gose, E. 1969. An evolutionary pattern recognition network. *IEEE Transactions on Systems Science and Cybernetics*, 5(3): 247–250.

Kostas, J.; Nota, C.; and Thomas, P. 2020. Asynchronous Coagent Networks. In *International Conference on Machine Learning*, 5426–5435. PMLR.

Lillicrap, T. P.; Santoro, A.; Marris, L.; Akerman, C. J.; and Hinton, G. 2020. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6): 335–346.

Mazzoni, P.; Andersen, R. A.; and Jordan, M. I. 1991. A more biologically plausible learning rule for neural networks. *Proceedings of the National Academy of Sciences*, 88(10): 4433–4437.

Movellan, J. R. 1991. Contrastive Hebbian learning in the continuous Hopfield model. In *Connectionist models*, 10–17. Elsevier.

Narendra, K. S.; and Thathachar, M. A. 1974. Learning automata-a survey. *IEEE Transactions on systems, man, and cybernetics*, (4): 323–334.

Narendra, K. S.; and Thathachar, M. A. 2012. *Learning automata: an introduction*. Courier corporation.

Nicoll, R. A. 2017. A brief history of long-term potentiation. *Neuron*, 93(2): 281–290.

O'Reilly, R. C. 1996. Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. *Neural computation*, 8(5): 895–938.

Pozzi, I.; Bohte, S.; and Roelfsema, P. 2020. Attention-Gated Brain Propagation: How the brain can implement reward-based error backpropagation. *Advances in Neural Information Processing Systems*, 33.

Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, 525–542. Springer.

Roelfsema, P. R.; and Ooyen, A. v. 2005. Attention-gated reinforcement learning of internal representations for classification. *Neural computation*, 17(10): 2176–2214.

Rombouts, J. O.; Bohte, S. M.; and Roelfsema, P. R. 2015. How attention can create synaptic tags for the learning of working memories in sequential tasks. *PLoS Comput Biol*, 11(3): e1004060.

Scellier, B.; and Bengio, Y. 2017. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11: 24.

Selfridge, O. G. 1988. Pandemonium: A paradigm for learning. In *Neurocomputing: Foundations of research*, 115–122. A Bradford Book.

Seung, H. S. 2003. Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. *Neuron*, 40(6): 1063–1073.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Thomas, P. S. 2011. Policy gradient coagent networks. In *Advances in Neural Information Processing Systems*, 1944–1952.

Thomas, P. S.; and Barto, A. G. 2011. Conjugate Markov Decision Processes. In *International Conference on Machine Learning*, 137–144.

Tsetlin, M. L.; et al. 1973. *Automaton theory and modeling of biological systems*, volume 102. Academic Press New York.

Uhr, L.; and Vossler, C. 1961. A pattern recognition program that generates, evaluates, and adjusts its own operators. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, 555–569.

Weber, T.; Heess, N.; Buesing, L.; and Silver, D. 2019. Credit assignment techniques in stochastic computation graphs. In *The 22nd International Conference on Artificial Intelligence and Statistics*, 2650–2660. PMLR.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4): 229–256.

Yin, P.; Lyu, J.; Zhang, S.; Osher, S.; Qi, Y.; and Xin, J. 2018. Understanding Straight-Through Estimator in Training Activation Quantized Neural Nets. In *International Conference on Learning Representations*.

Zhang, K.; Yang, Z.; and Başar, T. 2021. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, 321–384.