Multi Agent Meta Reinforcement Learning on Neural Networks

Alexander Cai, Oliver Cheng

December 13, 2022

1 Introduction to Meta-Reinforcement Learning

In recent years there has been a growing amount of excitement about *meta-learning* in order to solve a wider set of problems. This is often used in reinforcement learning tasks. In a standard reinforcement learning task, we have a policy that governs how an agent transitions between states in an environment. There are rewards that the agent can receive, and the optimal policy is one that maximizes rewards. We can formalize this by defining a reinforcement learning regime as a Markov Decision Process (MDP) where S is the set of states, A is the set of actions, T is the transition distribution (how states transition given an action), and R is the reward, which is a function of the state. Let π_θ be the policy, which is parameterized by some θ. Thus for some "task" $\mathcal{T}$, the goal is thus to maximize the objective

$$\mathcal{J}_\mathcal{T}(\theta) = \mathbb{E}_{a_t \sim \pi_\theta(s_t), s_{t+1} \sim T(s_t, a_t)} \left(\sum_t \gamma^t R(s_t, a_{t-1}) \right).$$

The task $\mathcal{T}$ determines the rewards, and hence the policy that is optimized depend heavily on what the task $\mathcal{T}$ is.

Reinforcement learning is used to solve many problems such as Go, Chess, Atari games, etc. where the environment is unknown and is structured in a way which an agent must discover an optimal strategy. This framework gives us a way to view human beings and the nervous system as a reinforcement learning program where we humans are the agent. (Un)Fortunately, this is a bit too basic of a way of modeling human beings and the brain, as reinforcement learning networks have not taken over humanity yet (citation needed). One of the reasons is that rewards and the task that an agent needs to optimize a policy for is constantly changing in the real world. Thus we introduce the idea of meta-learning, where rather than optimizing a single policy for a single task $\mathcal{T}$, we optimize for a learning strategy on how to optimize a policy for a distribution of tasks $p(\mathcal{T})$. This idea is motivated by the idea in psychology of "learning how to learn" and is conceptually closer to how the brain works (that one prefrontal cortex Deepind paper.). With meta-learning, due to the way we are improving the way we are finding the policy, this is a way we can introduce the idea of "past experiences" to help train policy for tasks that are similar.

2 Meta-Reinforcement Learning in the Brain

Basically try and repeat the thing in Box 4 of https://www.sciencedirect.com/science/article/pii/S136466131930061

maybe this is garbage lol

In the Theory of Neural Computation, we can see reinforcement-learning-esque regimes. In particular researchers have found that the dopamine system seems to follow the TD algorithm, a reinforcement learning algorithm. In particular, the error function of the TD algorithm operates in the same way to how dopamine neurons learn their firing rates in order to associate stimuli with future rewards. In this case, the dopamine neuron is the agent and it can change its firing rate as an action.

3 Implementation of Meta-Learning MARL

4 Comparison to Known Learning Rules

We compare this meta-learning (learning of the best learning rule) to accepted learning rules in the literature. Stochastic gradient descent (SGD) converges the quickest, but due to the weight transport problem, is not biologically plausible (citation needed). Biologically plausible alternatives involve learning from a global error signal, and involve perturbation-type learning rules. Global error signals have been observed in the brain, but it is still under much research exactly how the brain uses these error signals in order to learn and update neurons. We will look at node (NP) and weight perturbation (WP).

The idea behind node or weight perturbation is to take some perturbation ξ of either the weights or the nodes, and then see how that changes the objective or loss. The learning rule update then corresponds to scaling this perturbation by whether or not it increases or decreases the loss. Consider a general neural network with L layers and a data set $\{\mathbf{x}_i^{(0)}, \mathbf{y}_i\}_{i=1}^N$. Let $\mathbf{W}_\ell \in \mathbb{R}^{R_{\ell+1} \times R_\ell}$ where R_ℓ is the number of neurons in layer ℓ. We define the forward pass as

$$\mathbf{X}^{(\ell)} = \sigma\left(\mathbf{W}_{\ell-1}\mathbf{X}^{(\ell-1)\top}\right). \tag{1}$$

Where σ is some non-linearity such as ReLU. We use MSE loss

$$E = \frac{1}{2}\|\mathbf{X}^{(L)} - \mathbf{Y}\|^2.$$

In SGD, we use backpropagation to update the weight parameters by

$$\Delta_{SGD}\mathbf{W}_\ell = \eta\frac{\partial E}{\partial \mathbf{W}_\ell},$$

where the chain rule means we need to "transport" these values backwards through the neural network. On the other hand, in WP/NP, we have two forward passes. For the first, we define the same as 1, however for NP, we will let the perturbation be a vector $\xi \in \mathbb{R}^{R_\ell}$ where all entries are distributed i.i.d $\mathcal{N}(0, \sigma)$. We can then define the forward pass as

$$\widetilde{\mathbf{X}}^{(\ell)} = \sigma\left(\mathbf{W}_{\ell-1}\widetilde{\mathbf{X}}^{(\ell-1)\top} + \xi^{(\ell)}\right).$$

While for WP we take $\mathbf{\Xi}^{(\ell)} \in \mathbb{R}^{R_{\ell+1} \times R_\ell}$ where all entries are distributed i.i.d $\mathcal{N}(0, \sigma)$.

$$\hat{\mathbf{X}}^{(\ell)} = \sigma\left(\left(\mathbf{W}_{\ell-1} + \mathbf{\Xi}^{(\ell-1)}\right)\hat{\mathbf{X}}^{(\ell-1)\top}\right).$$

We can then define two errors for each NP and WP respectively.

$$E_N = \frac{1}{2}\|\widetilde{\mathbf{X}}^{(L)} - \mathbf{Y}\|^2 \qquad\qquad E_W = \frac{1}{2}\|\hat{\mathbf{X}}^{(L)} - \mathbf{Y}\|^2.$$

From here we can simply transmit the global error signal $E - E_N$ or $E - E_W$ and update our weights according to this error signal. In particular, we update parameters as follows for NP

$$\Delta_{NP}\mathbf{W}_\ell = \frac{\eta}{\sigma}(E - E_N)\sum_{i=1}^{R_{\ell+1}} \mathbf{x}_i^{(\ell)}\xi^{(\ell+1)\top}$$

While for WP we update as

$$\Delta_{WP}\mathbf{W}_\ell = \frac{\eta}{\sigma}(E - E_W)\mathbf{\Xi}^{(\ell)}$$

The idea is that perturbations which cause decrease the objective will be added to the parameters of the model. In expectation we have that up to a constant

$$\langle \Delta_{NP}\mathbf{W}_\ell \rangle_{\xi^{(\ell)}} = \langle \Delta_{WP}\mathbf{W}_\ell \rangle_{\mathbf{\Xi}^{(\ell)}} = \Delta_{SGD}\mathbf{W}_\ell.$$

Node and weight perturbation are often used in small teacher-student learning cases where there is only a single perceptron. Recent work by Hiratani et al. explores node perturbation in neural networks with a hidden layer, and finds computational and stability issues that suggest biological implausibility (cite hiratani et al). Nonetheless, these perturbation techniques remain an interesting way to investigate possible learning rules for neurons.