Backpropagation-Free Parallel Deep Reinforcement Learning

William H. Guss
Machine Learning at Berkeley
wguss@berkeley.edu

Phillip Kuznetsov
Machine Learning at Berkeley
Berkeley CA, 94720
philkuz@berkeley.edu

Noah Golmant
Machine Learning at Berkeley
Berkeley CA, 94720
noah.golmant@berkeley.edu

Max Johansen
Machine Learning at Berkeley
Berkeley CA, 94720
maxjohansen@berkeley.edu

Abstract

In this paper we conjecture that an agent, envirionment pair (μ, E) trained using DDPG with an actor network μ and critic network Q^{μ} can be decomposed into a number of sub-agent, sub-environment pairs (μ^n, E^n) ranging over every neuron in μ; that is, we show empirically that treating each neuron n as an agent $\mu^n : \mathbb{R}^n \to \mathbb{R}$ of its inputs and optimizing a value function Q^{μ^n} with respect to the weights of μ^n is dual to optimizing Q^{μ} with respect to the weights of μ. Finally we propose a learning rule which simultaneously optimizes each μ^n without error backpropogation.

1 Introduction

Recent techniques in deep reinforcement learning attempt to learn two deep neural networks: one to approximate a value function and one to learn a deterministic policy that maximizes this value function according to the action the agent performs.

Other work has attempted to address the issue of decoupling connections between layers in the network using decoupled neural interfaces [1]. Synthetic gradient modules model the error gradient using only local information, allowing immediate feedback that is later corrected when the backpropogated error is finally computed.

Both of these techniques still rely on biologically implausible backpropagation mechanisms to learn the parameters of the networks. We attempt to address these issues by decomposing the agent and learning rules at the local neuron level.

2 Agent-Environment Value Decomposition

In this section we develop a theoretical basis for decomposing the agent and its environment into local agents which take in prior neuron activations as state and output activations as the neuron's own actions. We will then show that under some mild conditions, these agents act in environments which are so simple that it suffices to estimate the policy gradient using a linear approximation. These results lead to a new local learning rule for deep reinforcement learning without the use of error-backpropagation.

2.1 Background

Recall the standard reinforcement learning setup. We say E is an *environment* if $E \overset{\text{def}}{=} (\mathcal{S}, \mathcal{A}, \mathcal{R}, T, r)$ where T describes transition probability measure $T(s_{t+1} \mid s_t, a_t)$ and $r : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ is a reward function. Furthermore $\mathcal{S}$, $\mathcal{A}$, $\mathcal{R}$ are the *state space, action space, and reward space* respectively. We restrict $\mathcal{R}$ to a compact subset of $\mathbb{R}$ and action space and state space to finite dimensional real vector spaces. As in DDPG [2] we assume that the environment E is *fully observed*; that is, at any time step the state s_t is fully described by the observation presented, x_t, and not by the history $(x_1, a_1, \ldots, a_{t-1})$.

We define the policy for an agent to be $\mu : \mathcal{P}(\mathcal{A}) \times \mathcal{S} \to [0, 1]$. We will deal only with *deterministic* policies where for every s_t there is unique a_t so that $\mu(\{a_t\} \mid s = s_t) = 1$ and the measure is 0 otherwise. Thus we define a *deterministic agent* by a policy function $\mu : \mathcal{S} \to \mathcal{A}$. Additionally we denote the state-space trajectories of μ by

$$\Gamma_\mu(\mathcal{S}) = \{((s_1, a_1), (s_2, a_2) \dots) \mid s_1 \sim T(s_0), s_{t+1} \sim T(s \mid s_t, \mu(s_t))\}. \tag{2.1.1}$$

For a policy μ the action-value function is the expected future reward under μ by performing a_t at state s_t. A temporally local definition thereof can be obtained using the Bellman equation

$$Q^\mu(s_t, a_t) = \mathop{\mathbb{E}}_{s_{t+1} \sim E} [r(s_t, a_t) + \gamma Q^\mu(s_{t+1}, \mu(s_{t+1}))] \tag{2.1.2}$$

with $\gamma \in (0, 1)$ a discount factor, and the second expectation removed because μ is deterministic.

The deep reinforcement learning approach to solving environments (MDPs) has predominately been separated into deterministic policy gradient methods and direct Q-Learning methods. In deterministic policy gradient (DPG) methods, we define an actor $\mu : \mathcal{S} \to \mathcal{A}$ and a critic $Q^\mu : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and optimize $Q^\mu(s_t, \mu(s_t|\theta^\mu))$ with respect to the paramers θ^μ of μ. Deep determinsitic policy gradient (DDPG) learning directly learns to approximate Q_a by creating two different deep neural networks for the actor and the critic and then back-propagating the Q gradient from the critic to the actor. Specifically, DDPG maximizes $Q^\mu(s_t, \mu(s_t|\theta^\mu))$ with respect to the paramers θ^μ of μ. This method is provably the true policy gradient of μ if Q^μ is known.

2.2 Towards Neurocomputational Decomposition of Q^μ

In order to decompose the Q^μ algorithm we will abstractly define a neurocomputational agent in terms of an operator on voltages with no restrictions on the topology of the network, and then relate the action-value function of the whole agent to those which are defined for each individual neuron in the network.

Definition 2.2.1. *If $\mathcal{V}$ is an N-dimensional vector space then a* ***neurocomputational agent*** *is a tuple $\mathcal{N} = (\mu, \epsilon, \delta, K, \Theta, \sigma, D)$ such that:*

- *$\epsilon : \mathcal{S} \to \mathcal{V}$ encodes the state into the voltages. Realistically, only a subset of all neurons are input neurons, denoted as $N_I \subset \mathcal{V}$, so $\epsilon(s_t) = \pi_{N_I}(\epsilon(s_t))$, where $\pi_K(x)$ is the cannonical projection of x on dimension(s) K,*
- *$\delta : \mathcal{V} \to \mathcal{A}$ decodes the voltages of the* output neurons *$N_O \subset V$ into an action, so that $\delta(v_t) = \delta(\pi_{N_O}(v_t))$.*
- *$K : \mathcal{V} \to \mathcal{V}$ is the linear voltage graph transition function of the graph representing the topolopy of $\mathcal{N}$, parameterized by θ.*
- *$\Theta : \mathcal{V} \to \mathcal{V}$ is a nonlinear inhibition function.*
- *$\sigma : \mathcal{V} \to \mathcal{V}$ is the elementwise application of some activation function to the voltage vector.*
- *$D : \mathcal{V} \times \mathcal{V} \to \mathcal{V}$ is called voltage dynamic of $\mathcal{N}$ such that*

  $$v_{t+1} \overset{def}{=} D(v_t, v_{in}) \overset{def}{=} \sigma(\Theta K[v_t]) + v_{in} \tag{2.2.1}$$

  where v_t is the internal voltage vector of $\mathcal{N}$ at time t and v_{in} is an input voltage to the network. We will occasionally abuse notation an say that $D(v_t) = D(v_t, 0)$ when v_{in} is 0.
- *$\mu : \mathcal{S} \to \mathcal{A}$ is the deterministic policy for $\mathcal{N}$ such that*

  $$\mu(s_t) = \delta(D(v_t, \epsilon(s_t)) \tag{2.2.2}$$

This definition encompasses any DQN or DDPG network with either reccurent or feedforward layers. In this paper we will mainly discuss standard feed forward neural networks, in which case Θ is not defined and the neural dynamics are repeatedly applied until an output of N_O is produced.

Definition 2.2.2. *If n is some neuron in $\mathcal{N}$, we say $E^n = (\mathcal{S}, \mathcal{A}, \mathcal{R}, T^n, r^n)$ is a deterministic* ***sub-environment*** *of E with respect to $\mathcal{N}$ if*

- *The state space is $\mathcal{S} = \mathcal{V}$; that is, the environment that neuron n observes is the voltages of all other neurons. Although this definition permits a fully connected graph for $\mathcal{N}$, realistically, each neuron only sees the voltages of a subset of all neurons. In the case of standard feed forward networks, n would observe only the activations of the previous layer.*
- *The action space is the set of voltages $\mathcal{A} = \mathbb{R}$ which the neuron n can output.*
- *The reward space $\mathcal{R}$ is the same as that of the base environment E. Furthermore the subenvironment emitts the same same reward as does the base environment. If the agent μ of $\mathcal{N}$ acts on a state s_t and receives a reward r_t, then every sub-environment emits the same reward r_t.*

- *The transition function $T : \mathcal{V} \times \mathbb{R} \to \mathcal{S}^n$ is such that*

$$T^n(v_t, \alpha_t) = (I - \chi_{n,n})D(v_t)) + e_n\alpha + \epsilon(s_t) \tag{2.2.3}$$

  where e_n is the n^{th} unit basis vector, I is the identity, and $\chi_{n,n} = 1$ if $n = n$ and 0 otherwise. Intuitively, the transition in E^n is the normal[1] neural dynamics on $\mathcal{N}$ except for at the neuron n, itself; in E^n we set the voltage of n in v_{t+1} to be the voltage chosen, α_t, plus the newly encoded voltage $\epsilon(s_t)$.

Lastly an agent $\mu^n : \mathcal{V} \to \mathbb{R}$ is called ***neuromorphically local*** *to $\mathcal{N}$ if $\mu^n : v_t \mapsto \langle D(v_t), e_n\rangle$; that is, μ^n acts according to the normal dynamics.*

With the basic definitions given, we show how reinforcement learning on $\mathcal{N}$ can be decomposed into the dual problem of local reinforcement learning on agents and environments of type given in Definition 2.2.2.

Theorem 2.2.3 (Neurocomputational Decompostion)**.** *Let E be an environment and $\mathcal{N}$ be a neurocomputational agent. Then there exists a set of agent environment pairs $\mathfrak{D}_{\mathcal{N}} \overset{def}{=} \{(E^n, \mu^n)\}_{n\in\mathcal{N}}$ such that for every $n \in \mathcal{N}$, the following diagram commutes*

$$\begin{tikzcd}
\mathcal{V} \times \mathcal{S} \arrow[r, "\mu\circ\pi_2"] \arrow[d, "\pi_1\times\epsilon\circ\pi_2"] & \mathcal{A} \\
\overbrace{\mathcal{V} \times \mathcal{V}}^{(v_t,\epsilon(s_t))} \arrow[r, "D"] \arrow[dr, "\mu^n\circ\pi_1+\pi_n\circ\pi_2"'] & \overbrace{\mathcal{V}}^{v_{t+1}} \arrow[u, "\delta"'] \arrow[d, "\pi_n"] \\
 & \mathbb{R}
\end{tikzcd} \tag{2.2.4}$$

Proof in Appendix

Theorem 2.2.3 gives a natural connection between the state space trajectories of μ and μ^n because in $\mathcal{N}$, the voltage v_t is a hidden variable which governs action of $\mathcal{N}$ at any timestep and dually the state s_t is a hidden variable of the Markov Decision Process formed by E^n which governs the state given by T^n at any timestep.

Naturally, the following question arises: does DPG learning on $\mathcal{N}$, specifically μ on E, commute with performing DPG learning on on every $(E^n, \mu^n) \in \mathfrak{D}_{\mathcal{N}}$? Supposing that we have the true Q^μ function and μ is optimal with respect to Q^μ, then it is intuitive, but not obvious, that every μ^n should behave optimally with respect to an Q^{μ^n} – but will the reverse hold? To answer these questions we give the following result.

Theorem 2.2.4. *If $\mathcal{N}$ is a nuerocomputational agent in E, then policy gradient for μ agrees with the simultaneous policy gradients of its decomposition; that is for every $(E^n, \mu^n) \in \mathfrak{D}_{\mathcal{N}}$*

$$\nabla_{K^n} Q^{\mu^n}(v, \alpha)\Big|_{v=v_t,\alpha=\mu^n(v_t)} = \nabla_{K^n} Q^{\mu}(s, a)\Big|_{s=s_t,a=\mu(s_t)} \tag{2.2.5}$$

for every time step t, where K^n is the nth column of the linear voltage graph transition matrix, i.e. the weights of the connections from all neurons to neuron n.

Proof in Appendix

Remarkably, the prevous theorem shows that optimizing every neuromorphically local agent μ^n with respect to a critic is exactly equivalent to optimizing the entire $\mathcal{N}$ with respect to a global critic Q^μ. Additionally optimization of the each μ^n does not require any backpropagation of $\nabla_a Q^{\mu^n}$ through the network since every Q^{μ^n} is diagramatically independent of eachother! As a result the neurocomputational decomposition $\mathfrak{D}_{\mathcal{N}}$ of $\mathcal{N}$ can be trained asynchronously and simultaneously.

2.2.1 Simplicity of Sub-Environments

An algorithm which learns $\mathcal{N}$ by performing DPG learning on every $(E^n, \mu^n) \in \mathfrak{D}_{\mathcal{N}}$ is only beneficial if each Q^{μ^n} can be approximated simply. For example if Q^{μ^n} are so complex that an approximation parameterized by K requires more than $|K^n| \notin O(1)$ parameters, then standard error-backpropagation is sureley superior.

However the results established in [1] and [3] show that the approximation of the error backpropagated signal can be linear in the activations. In the framework of DDPG, this intuitively says that the infintesmal effect of the weights K^n on the dynamics and thereby the true Q^μ of a neurocomputational agent $\mathcal{N}$ might be linear in the local neural neighborhood of n.

[1]Since the state space of E^n is only $\mathcal{V}$, s_t is a hidden variable of the markov decision process, and T should be a function on $\mathcal{V} \times \mathcal{S} \times \mathbb{R}$, but this is ommitted for simplicity.

3 Decentralized Deep Determinstic Policy Gradient Learning

We use the preceding hypothesis as a motivation for the following local decomposition. We decompose the actor-critic dual networks of DDPG into a set of actor-critic pairs, where each actor neuron attempts to learn a local policy to maximize the value of its own critic. This critic is a linear regressor and its gradient agrees with the gradient of a global Q network when the policy is *compatible* as in the framework of [3].

4 Experimentation

4.1 Experiment 1: Learning Q functions on components of the Actor network.

Motivation: Our first batch of experiments aimed to establish empirically that the Q-functions of each layer, denoted $Q_1 ... Q_L$ are similar to the Q-function of the overall network, denoted Q_μ. To that end, we extended the actor-critic methodology used in Lillicrap et al (2016) as follows. In addition to training Q^μ, to estimate the Q functions of the actor network μ, sub-critic networks Q^n were initialized for each individual layer. We then compare the Q values estimated by the subcritics to the Q estimation provided by the main critic, Q^μ.

To determine and compare the rate at which the subcritics and the main critic learn, the experiment was run in two phases. First, each of the subcritics and the main critic were trained using the standard DDPG algorithm on some actor μ. In the second phase, a new actor μ' was initialized and its Q-function set to Q_μ as determined above in phase 1. The subcritics were trained, and the values of $Q_1 \dots Q_L$ were plotted as training occurred.

4.2 Experiment 2: Treating each neuron as its Actor-Critic network using linear approximators

Now that we have established that the Critic networks for each of the individual neuron learns the same Q-function as does the entire agent, we now treat each neuron as its own actor. Each neuron changes the weights of its presynaptic neurons' connections, as parameters for its actor – its voltage on the next timestep – to optimize its learned approximated Q function. We use the linear approximation:

$$Q^n(v, a) \approx \theta_{n,v}^T v + \theta_{n,a} a = \theta_n^T (v, a)^T$$
$$\mu^n(v) = \sigma(K^n v)$$

Intuitively, it should be the case that by treating each neuron in the neural network as its own Q-learner, the policy gradient of an individual agent should be the same as for the policy gradient entire agent, if each neuron sees the same reward r_t as the entire agent. In other words, the optimal way of updating the entire connection matrix K should be the optimal way for updating the weights connected to each neuron w.r.t. each neuron as its own Q-learner, given the same rewards.

5 Results

5.1 Experiment 1

We use the Pearson correlation coefficient to determine the similarity between the derivatives of the global Q function critic and the local approximations we use a-la DDPG.

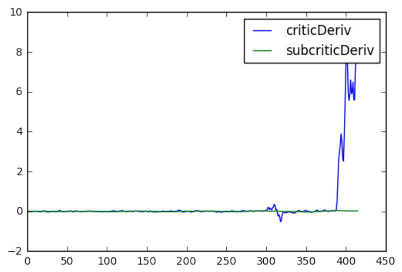
We wish to approximate this global Q using less expensive local Q approximator functions. We can ignore the magnitude and actual values of the two functions as we only wish to examine the updates, i.e. derivatives of these functions. Thus, we attempt to **find a positive correlation between $\frac{\partial Q_{global}}{\partial t}$ and $\frac{\partial Q_{local}}{\partial t}$.**

This table can be used to roughly understand the importance of correlations between two functions.

In the mountain car experiment, we find a Pearson r-value of 0.389. Thus, we can infer that there is a medium association between the derivatives of the global and local function.

5.2 Experiment 2

The second experiment implemented the local training paradigm suggested above. We tested our results with a simple 1D synthetic environment where the agent was given 100 reward for stepping on a block 100 steps to the right, a -1 reward for moving left and 0 reward for all positions between the starting block and the 100 reward block.



(a) Critic and subcritic derivatives

	Coefficient, r	
Strength of Association	Positive	Negative
Small	.1 to .3	-0.1 to -0.3
Medium	.3 to .5	-0.3 to -0.5
Large	.5 to 1.0	-0.5 to -1.0

(b) Pearson interpretation

Figure 1: Analysis figures

The agent easily learned this environment using a two layer network. To generalize our results, we moved onto a more difficult environment, Mountain Car. This environment requires an agent to drive a car up a mountain.

However, the car doesn't have enough power to make it up the mountain and must first drive back, the direction opposite the goal, then drive forward to successfully complete the environment.

We successfully solved the environment with small simple networks, however, if we tried to use multiple layers the local Q functions began to diverge. We believe this is a result of the high variance in the local environments of each neuron as a result of constantly feeding in values and will require more evaluation to correct this issue.

6 Conclusion

The results of experiments 1 and 2 show that local learning rules are plausible. We were able to teach a network to perform a simple task in OpenAI gym using a small network structure. Furthermore, we should that we could employ local *linear* learning rules to reach this end. This is shown in our results of experiment 2.

6.1 Future Work

We want to scale our results to include truly deep neural network architectures. We show promising results using multi-layer networks, however we quickly fall victim to issues from high variance in large networks.

There are a number of techniques deep learning practitioners use to limit variance in standard deep learning architectures.

One technique we're curious about is probabilistically fixing a subset of the neurons, while allowing the remaining neurons to *explore*. This technique is very similar to the dropout used in deep learning research. We believe this will also give us an opportunity to leverage the gains of implicit ensembling that is gained through traditional dropout techniques.

7 Appendix

Proof of (2.2.4)

Proof. If E and $\mathcal{N}$ are given, then for every $n \in \mathcal{N}$ let E^n be a sub-environment of E with respect to $\mathcal{N}$. Next let μ^n be a neuromorphically local agent in E^n with respect to $\mathcal{N}$.

We first show that (2.2.4) commutes. Let $v_t \in \mathcal{V}$ and $s_t \in \mathcal{S}$. Observe that

$$[\mu \circ \pi_2](v_t, s_t) = \mu(s_t) = \delta\left(D(v_t, \epsilon(s_t))\right) = [\delta \circ D \circ (\pi_1 \times \epsilon \circ \pi_2)](v_t, s_t),$$

and therefore the top half of the diagram commutes. Given some $(v_t, \epsilon(s_t)) \in \mathcal{V} \times \mathcal{V}$ we have that

$$\begin{aligned}[\mu^n \circ \pi_1 + \pi_2](v_t, \epsilon(s_t)) &= \mu^n(v_t) + \pi_n(\epsilon(s_t)),\\ &= \pi_n\left(D(v_t) + \epsilon(s_t)\right),\\ &= [\pi_n \circ D](v_t, \epsilon(s_t)).\end{aligned}$$

Thus the diagram in (2.2.4) commutes. ☐

Proof of (2.2.5)

Proof. Let $\mathcal{N}, E$ be given and fix $(E^n, \mu^n) \in \mathfrak{D}_{\mathcal{N}}$. For any initial state s_0, Theorem 2.2.2 gives that the state-action trajectory $\kappa \in \Gamma_\mu(\mathcal{S})$ generated by μ from s_0 is dual to the state-action trajectory $\kappa^* \in \Gamma_{\mu^n}(\mathcal{V})$ generated by μ^n from $v_0 = 0$ when the hidden state of T^n is s_t. Because E^n is a sub-environment of E, $r^n \equiv r$ the sequence of rewards on κ and κ^* are the same. Using the definition of the action-value function assuming that s_0 fixed,

$$Q^\mu(s_t, a_t) = \sum_{\tau=t}^{\infty} r(s_t, \mu(s_t))\gamma^{\tau-t} = \sum_{\tau=t}^{\infty} r^n(v_t, \mu^n(v_t))\gamma^{\tau-t} = Q^{\mu^n}(v_t, a_t) \tag{7.0.1}$$

where $\kappa = ((s_t, \mu(s_t)))_{t\in\mathbb{N}}$ and $\kappa^* = ((v_t, \mu^n(v_t)))_{t\in\mathbb{N}}$.

If (v_t, s_t) are give, Theorem 2.2.3 states that both μ^n and μ commute with the dynamics on $(v_t, \epsilon(s_t))$ (see the middle path in (2.2.4)). Thus if the dynamics are parameterized by $K^n \in \mathcal{K}_n$, application of the equality in (7.0.1), gives the following commutitive diagram, $\mathfrak{k}_n$:

$$\begin{tikzcd}
 & \mathcal{A} \arrow[dr, "Q^\mu(s_t,\cdot)"] & \\
\mathcal{K}_n \arrow[ur, "\mu(s_t)"] \arrow[r, "D(v_t,\epsilon(s_t))"] \arrow[dr, "\mu^n(v_t)"'] & \mathcal{V} \arrow[u, "\delta"'] \arrow[d, "\pi_n"'] & \mathbb{R} \\
 & \mathbb{R} \arrow[ur, "Q^{\mu^n}(v_t,\cdot)"'] &
\end{tikzcd} \tag{7.0.2}$$

Recall from category theory that differentiation is a functor on the category of C^1 manifolds $\mathbf{Man}^1$ because for any two morphisms $f, g \in \mathrm{Hom}(\mathbf{Man}^1)$, $\nabla f \circ g = \nabla f \circ \nabla g$ by chain rule. It then follows that the diagram $\nabla(\mathfrak{k}_n)$ commutes and so

$$\begin{aligned}\nabla_{K^n} Q^{\mu^n}(v_t, \alpha)\Big|_{\alpha=\mu^n(v_t)} &= \nabla_\alpha Q^{\mu^n}(v_t, \alpha)\nabla_{K^n}\mu^n(v_t)\\ &= \nabla_a Q^\mu(s_t, a)\nabla_{K^n}\mu(s_t)\\ &= \nabla_{K^n} Q^\mu(s_t, a)\Big|_{a=\mu(s_t)}\end{aligned}$$

Because this equality holds for any (s_t, v_t) and therefore any state-action trajectory with arbitrary initial hidden variables, the theorem holds. ☐

References

[1] Max Jaderberg et al. "Decoupled Neural Interfaces using Synthetic Gradients". In: *CoRR* abs/1608.05343 (2016). URL: `http://arxiv.org/abs/1608.05343`.

[2] Timothy P. Lillicrap et al. "Continuous control with deep reinforcement learning". In: *CoRR* abs/1509.02971 (2015). URL: `http://arxiv.org/abs/1509.02971`.

[3] Philip S Thomas. "Policy gradient coagent networks". In: *Advances in Neural Information Processing Systems*. 2011, pp. 1944–1952.