CS861 Final Paper: A Survey of Policy Gradient Algorithms

Changyu Gao **Zihao Chen**

Abstract

In contrast to value-based methods, policy gradient methods directly model the policy and perform gradient updates. We review the basics of policy gradient and cover classical algorithms of actor-critic methods. Specific algorithms covered in this survey include Simple Actor-Critic, Trust Region Policy Optimization, Proximal Policy Optimization, synchronous Advantage Actor-Critic, and Deterministic policy gradient. Important results are presented with sketched proofs.

1 Introduction

The reinforcement learning framework has gained increasing attention in recent years. There are two major types of algorithms used in reinforcement learning [Sutton and Barto, 2018] – value-based methods and policy gradient methods. Specifically, value-based methods first learn the values of actions and then generate policies according to the values, while policy gradient methods directly optimize the policy without explicitly estimating the values.

The paper will focus on the policy gradient methods with an analysis component. We will start by explaining the key concepts: tabular methods vs approximation methods, Q-learning. We then give an overview of policy gradient methods and sketched a proof of the policy gradient theorem. After that, various algorithms will be covered.

2 Background Knowledge

2.1 Basics concepts and notations

State: s, S, S_t; Agent: a, A, A_t; Rewards: r, R, R_t

γ: Discount factor; penalty to uncertainty $(0.1]$

$\pi_\theta(a|s)$: Stochastic policy parameterized by θ

$V(s), V^\pi(s)$: State value function

$Q(s,a), Q^\pi(s,a)$: Action value function

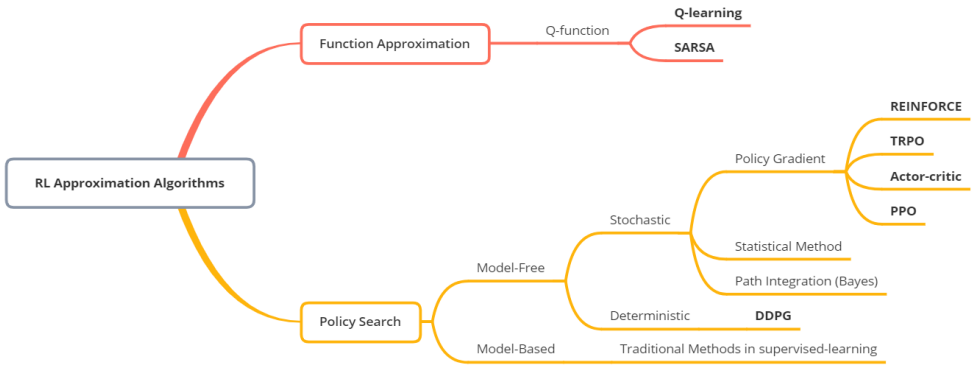
$A(s,a)$: Advantage function,$A(s,a) = Q(s,a) - V(s)$

2.2 Tabular Methods

Tabular methods refer to problems that the state and action spaces are small enough to be represented as tables. This paper focuses on the approximation methods.

2.3 Approximation Methods

Approximation methods are usually applied to problems with arbitrarily large spaces. Function approximation and Policy search are two mean methods under the approximation methods. This paper will focus on the latter.



2.4 Q-learning

Q-learning is a model-free TD control algorithm. It is designed for MDPs with finite states. Its update formula is

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

where α is the learning rate. Intuitively, we are using the current reward plus the current best estimate for the next state $r_t + \gamma \max_a Q_a(s_{t+1}, a)$ to correct current action-value $Q(s, a)$. The update does not depend on the next state, so the algorithm is off-policy. On the other hand, we can choose the action per the policy derived from the current Q-function and use the chosen action to correct current estimate, making it an on-policy algorithm named State–action–reward–state–action (SARSA). The SARSA update is given as follows

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

where a_{t+1} is chosen by some policy derived from current $Q(s, a)$.

To handle MDP problems with infinite states , we can use other value-based approaches. One popular choice is to use a neural network (Deep Q-Network, DQN) to parameterize $Q_w(s, a)$ and optimize w to match Q_w with the actual Q-function. We will see later such parameterization can be combined with policy gradient methods.

3 Policy Gradient

Policy gradient methods directly model on the policy. We parameterize the policy by θ and optimize θ with respect to J_θ, the expected reward under policy π_θ

$$\begin{aligned} J(\theta) &= \mathbb{E}_{\pi_\theta} \\ &= \sum_a \pi_\theta(a|s) Q^\pi(s, a) \end{aligned}$$

3.1 Difference between Q-learning and Policy Gradient

Q-learning and Policy Gradient are two main algorithms for reinforcement learning. Different from Q-learning which is based on function approximation, Policy Gradient is based on searching policy. In terms of supervised learning, Q-learning is similar to naive Bayes which calculate the post probability,

while Policy Gradient is similar to SVM that doesn't depend on post probability. Therefore, Policy Gradient will converge much faster than the traditional Q-learning algorithm.

However, in discrete cases, Q-learning can theoretically converge to the best solution, but gradient methods can only converge to the local maximum.

3.2 Policy Gradient Theorem

Calculating the gradient $\nabla_\theta J(\theta)$ is the most important step in the algorithm. It depends on both the Q-function and the policy probability, which adds difficulty to our computation. The Policy Gradient Theorem help us simplify the calculation of derivative. The theorem states

$$\begin{aligned}
&\nabla_\theta J(\theta) \\
=&\nabla_\theta \sum_s d^\pi(s) \sum_a Q^\pi(s,a)\pi_\theta(a|s) \\
\propto&\sum_s d^\pi(s) \sum_a Q^\pi(s,a)\nabla_\theta \pi_\theta(a|s)
\end{aligned}$$

where $d^\pi(s)$ is the stationary distribution of Markov chain for π_θ, $J(\theta)$ is our reward function. *Sketch Proof.* First, calculate the derivative of the state value function

$$\begin{aligned}
&\nabla_\theta V^\pi(s) \\
=&\nabla_\theta(\sum_a \pi_\theta(a|s)Q^\pi(s,a)) \\
=&\sum_a(\nabla_\theta\pi_\theta(a|s) + \pi_\theta(a|s)\nabla_\theta Q^\pi(s,a) \\
=&\dots \\
=&\sum_a(\nabla_\theta\pi_\theta(a|s) + \pi_\theta(a|s)\sum_{s'} P(s'|s,a)\nabla_\theta V^\pi(s'))
\end{aligned}$$

Since it is a recursive term, we can use the sum to represent $\nabla_\theta V^\pi(s)$. Define $\rho^\pi(s \to x, k)$ as the sequence of transitioning from state s to x in k steps with policy π_θ. Also, define $\phi(s) = \sum_a \nabla_\theta \pi_\theta(a|s)Q^\pi(s,a)$

$$\begin{aligned}
&\nabla_\theta V^\pi(s) \\
=&\phi(s) + \sum_a \pi_\theta(a|s)\sum_{s'} P(s'|s,a)\nabla_\theta V^\pi(s') \\
=&\phi(s) + \sum_{s'} \rho^\pi(s \to s', 1)\nabla_\theta V^\pi(s') \\
=&\phi(s) + \sum_{s'} \rho^\pi(s \to s', 1)[\phi(s') + \sum_{s''} \rho^\pi(s' \to s'', 1)\nabla_\theta V^\pi(s'')] \\
=&\dots \text{expand recursively} \\
=&\sum_{x\in S}\sum_{k=0}^{\infty} \rho^\pi(s \to, k)\phi(x)
\end{aligned}$$

Now, there is no derivative of Q-value function in the equation. Then plug into the reward function, we get

$$
\begin{aligned}
\nabla_\theta J(\theta) =& \nabla_\theta V^\pi(s_0) \\
=& \sum_s \sum_{k=0}^{\infty} \rho^\pi(s_0 \to s.k)\phi(s) \\
=& \sum_s \mu(s)\phi(s) \\
=& (\sum_s \mu(s)) \sum_s \frac{\mu(s)}{\sum_s \mu(s)}\phi(s) \\
\propto& \sum_s \frac{\mu(s)}{\sum_s \mu(s)}\phi(s) \\
=& \sum_s d^\pi(s) \sum_a \nabla_\theta \pi_\theta(a|s) Q^\pi(s,a)
\end{aligned}
$$

In different cases, policy gradient can be written in different forms. For further reading in these forms, please check the general advantage estimation paper [Schulman et al., 2018]. Here, we only discuss the base case.

3.3 Vanilla Policy Gradient Algorithm

In policy gradient methods, policies are parameterized by some parameter that we can optimize over. The central question becomes how we update the policy parameter. We will introduce the first policy-gradient learning algorithm REINFORCE [Williams, 1992]. It relies on Monte-Carlo estimates of the gradient to update policy parameters. REINFORCE can be generalized with a baseline, which may help reduce the variance and thus speed up the learning. A natural choice of the baseline is an estimate of the state value, which can be obtained along with the gradient update.

3.4 Policy Gradient with Q-Function Approximation

To perform policy gradient steps, we need to estimate $Q^{\pi_\theta}(s,a)$. In the simplest case, we can use Monte-Carlo simulation, possibly resulting in high variance. We can reduce the variance by using an approximation of the Q-function such as neural networks which are able to generalize. Let $f_w(s,a)$ be such an approximation parameterized by w. Assuming f is differentiable with respect to w, we naturally match f with Q and get the following update rule

$$
\begin{aligned}
\Delta w \propto& -\nabla_w \left\{ \frac{1}{2} \mathbb{E}_\pi \left[(f_w(s,a) - Q^\pi(s,a))^2 \right] \right\} \\
=& -\mathbb{E}_\pi \left[(f_w(s,a) - Q^\pi(s,a)) \nabla_w f_w(s,a) \right].
\end{aligned}
$$

When the learning process converges, we will have $\Delta w = 0$, so

If in addition f_w is compatible with the parameterized policy, then we can use f_w in replace of Q^π:

(3.1)

$$
\mathbb{E}_{\pi_\theta} \left[(f_w(s,a) - Q^\pi(s,a)) \nabla_w f_w(s,a) \right] = 0
$$

Theorem. *[Sutton 1999] If f_w satisfies (3.1) and is compatible with the policy, i.e.*

(3.2)

$$
\nabla_w f_w(s,a) = \frac{1}{\pi_\theta(a|s)} \nabla_\theta \pi_\theta(a|s) = \nabla_\theta \log \pi_\theta(a|s).
$$

Then the policy gradient is exact when using f_w in place of Q^π,

$$
\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[f_w(s,a) \nabla_\theta \log \pi_\theta(a|s) \right]
$$

Proof. From equations (3.1) and (3.2), we have

$$
\mathbb{E}_{\pi_\theta} \left[(f_w(s,a) - Q^\pi(a|s)) \nabla_w f_w(s,a) \right] = \mathbb{E}_{\pi_\theta} \left[(f_w(s,a) - Q^\pi(a|s)) \nabla_\theta \log \pi_\theta(s,a) \right] = 0
$$

It follows from policy gradient theorem that,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[Q^\pi(s,a)\nabla_\theta \log \pi_\theta(s,a)\right] = \mathbb{E}_{\pi_\theta}\left[f_w(s,a)\nabla_\theta \log \pi_\theta(s,a)\right].$$

□

3.5 Baseline and the Advantage Function

We can modify the gradient step by subtracting any baseline function $B(s)$ since

$$\mathbb{E}_{\pi_\theta}\left[B(s)\log \pi_\theta(s,a)\right] = \sum_{s\in\mathcal{S}} d^{\pi_\theta} B(s)\nabla_\theta \sum_{a\in\mathcal{A}} \pi_\theta(s,a) = 0,$$

where the last step is due to $\sum_{a\in\mathcal{A}} \pi_\theta(s,a) = 1$ for any s. An obvious candidate for a baseline is the state value function $V^{\pi_\theta}(s)$.

We define the advantage function A^{π_θ} by

$$A^{\pi_\theta}(s,a) = Q^\pi(s,a) - V^{\pi_\theta}(s).$$

And the modified policy gradient is given by

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[A^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(s,a)\right].$$

The variance can be greatly reduced when an appropriate baseline is used.

4 Actor-critic

As we have shown above, we can approximate $Q^{\pi_\theta}(s,a)$ by properly chosen $f_w(s,a)$. If we learn such f_w and perform the policy gradient simultaneously, we obtain the so-called Actor-critic algorithms. Two sets of parameters are updated in the process:

- **Critic**: Updates w to match $f_w(s,a)$ with $Q^{\pi_\theta}(s,a)$.
- **Actor**: Performs the policy gradient step — updates θ according to $f_w(s,a)$.

If the advantage function is used instead, the baseline function can be updated accordingly at the same time. As another improvement, we can use TD-learning to estimate $f_w(s,a)$. Putting all of this together, the process is described as follows

Algorithm 1 Simple Actor-Critic

Initialize θ and w.
Initialize s and sample $a \sim \pi_\theta(a|s)$
for each step **do**
  Observe r, s'
  Sample $a' \sim \pi_\theta(a'|s')$
  $\theta \leftarrow \theta + \alpha f_w(s,a)\nabla_\theta \log \pi_\theta(s,a)$
  $\delta \leftarrow r + \gamma f_w(s',a') - f_w(s,a)$
  $w \leftarrow w + \beta\delta\nabla_w f_w(s,a)$
  $a, s \leftarrow a', s'$
end for

where α and β are learning rates. If the task is episodic, we need to run the for-loop for multiple episodes. Regarding the convergence, Sutton has proved one version of convergence guarantees for infinite horizon MDPs.

5 Trust Region Policy Optimization

Basic policy gradient methods may suffer from dramatic changes of the parameters in one step. To alleviate this, we impose a constraint on the size of policy change. The trust region policy optimization (TRPO) algorithm [Schulman et al., 2017a] uses KL divergence as regularization. A simpler algorithm with similar performance is proximal policy optimization (PPO) [Schulman et al., 2017b], which uses a clipped surrogate objective instead.

5.1 The Procedure of TRPO

$$\theta_{new} = \theta_{old} + \alpha \nabla_\theta J$$

Determining the learning rate (α) is an important task for all algorithms based on gradient. TRPO is designed for finding an appropriate α so that the reward function $\eta(\pi)$ does not decrease. (The whole procedure and proof is too long, we only give a dense one here)

We use τ to represent a series of state and action $s_0, a_0, \ldots, s_H, a_H$, then η is:

$$\eta(\tilde{\pi}) = E_{\tau|\tilde{\pi}}[\sum_{t=0}^{\infty} \gamma^t (r(s_t))]$$

$\tilde{\pi}$ is the new policy.

A natural method will be split $\eta(\tilde{\pi})$ into reward function ($\eta(\pi)$) and other terms. By [Kakade and Langford, 2002], we have the following equation:

(5.1)

$$\eta(\tilde{\pi}) = \eta(\pi) + E_{s,a,\tilde{\pi}}[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t)] (5.1)$$

where π is the old policy, and $\tilde{\pi}$ is the new policy.

The advantage function is defined by

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s) = E_{s' P(s'|s,a)}[r(s) + \gamma V^\pi(s') - V^\pi(s)]$$

Proof for (5.1):

$$\begin{aligned}
& E_{\tau|\tilde{\pi}}[\sum_{0}^{\infty} \gamma^t A_\pi(s_t, a_t)] \\
=& E_{\tau|\tilde{\pi}}[\sum_{0}^{\infty} \gamma^t (r(s) + \gamma V^\pi(s_{t+1} - V^\pi(s_t))] \\
=& E_{\tau|\tilde{\pi}}[\sum_{0}^{\infty} \gamma^t (r(s_t)) + \sum_{0}^{\infty} \gamma^t (\gamma V^\pi(s_{t+1}) - V^\pi(s_t))] \\
=& E_{\tau|\tilde{\pi}}[\sum_{0}^{\infty} \gamma^t (r(s_t))] + E_{s_0}[-V^\pi(s_0)] \\
=& \eta(\tilde{\pi}) - \eta(\pi)
\end{aligned}$$

After we get (5.1), we need to get the policy term in our equation. Then the expectation of advantage function can be written as followed:

(5.2)

$$\eta(\tilde{\pi}) = \eta(\pi) + \sum_{0}^{\infty} \sum_{s} P(s_t = s|\tilde{\pi}) \sum_{a} \tilde{\pi}(a|s) \gamma^t A_\pi(s,a)$$

where $P(s_t = s|\tilde{\pi})\tilde{\pi}(a|s)$ is the joint distribution of (s,a); $\sum_a \tilde{\pi}(a|s)\gamma^t A_\pi(s,a)$ is the marginal distribution of a; $\sum_s P(s_t = s|\tilde{\pi})$ is the marginal distribution of s; The $\sum_0^\infty$ is summing up the whole time series.

Further, we define $\rho_\pi(s) = P(s_0 = s) + \gamma P(s_1 = s) + \gamma^2 P(s_2 = s) + \ldots$. Then (5.2) can be rewritten as

(5.3)

$$\eta(\tilde{\pi}) = \eta(\pi) + \sum_{s} \rho_{\tilde{\pi}}(s) \sum_{a} \tilde{\pi}(a|s) A_\pi(s,a)$$

Now, the distribution of s is generated by our new policy, but the weight for new policy is too large. From this step, we are dealing with the state distribution. First, we omit the difference between the states, using the old policy instead. When the old and new parameters are very close to each other, we can also use the old state instead. Then original cost function becomes:

(5.4)

$$L_\pi = \eta(\pi) + \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) A_\pi(s,a)$$

Since our new policy $\tilde{\pi}$ is parameterized by θ which is unknown, we cannot get the action. Therefore, we process the action distribution by their weight. Meanwhile, replace $\sum_s \rho_{\theta_{old}}(s)$ with $\frac{1}{1-\gamma} E_{s\rho_{\theta_{old}}}$. The surrogate reward function becomes:

(5.5)

$$L_\pi(\tilde{\pi}) = \eta(\pi) + E_{s\rho_{\theta_{old}}, a\pi_{\theta_{old}}}[\frac{\tilde{\pi}_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} A_{\theta_{old}}(s,a)]$$

Now compare (5.3) and (5.5); note that $L_\pi(\tilde{\pi})$ and $\eta(\tilde{\pi})$ are the same under the policy $\pi_{\theta_{old}}$:

(5.6)

$$L_{\pi_{\theta_{old}}}(\pi_{\theta_{old}}) = \eta(\pi_{\theta_{old}})$$
$$\nabla_\theta L_{\pi_{\theta_{old}}}(\pi_{old})|_{\theta=\theta_{old}} = \nabla_\theta \eta(\pi_\theta)|_{\theta=\theta_{old}}$$

Finally, we come to decide the learning rate α. We introduce KL Divergence for getting lower bound

(5.7)

$$\eta(\tilde{\pi}) \geq L_\pi(\tilde{\pi}) - c D_{KL}^{max}(\pi, \tilde{\pi})$$
$$c = \frac{2\epsilon\gamma}{(1-\gamma)^2}$$

The inequality gives the lower bound of $\eta(\tilde{\pi})$ and defined as

$$M_i(\pi) = L_{\pi_i}(\pi) - c D_{KL}^{max}(\pi_i, \pi)$$

We can use this lower bound to prove the monotone decreasing of our policy: With $\eta(\pi_{i+1}) \geq M_i(\pi_{i+1}), \eta(\pi)i) =<_i (\pi_+ i)$, we get $\eta(\pi_{i+}) - \eta(\pi_i) \geq M_i(\pi_{i+1}) - M(\pi_i)$. It remains to find the π_{i+1} to maximize the M_i so that η will be monotonously increasing. It becomes solving:

$$\text{maximize}[L_{\theta_{old}}(\theta) - c D_{KL}^{max}(\theta_{old}, \theta)$$

Taking advantage of the small penalty c, it becomes

$$\text{maximize}_\theta E_{s\rho_{\theta_{old}}, a\pi_{\theta_{old}}}[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} A_{\theta_{old}}(s,a)]$$

subject to

$$D_{KL}^{max}(\theta_{old}, \theta) \leq \delta$$

Since $|S| = \infty$, we use mean KL divergence to substitute the max KL divergence, and finally turns to:

$$\text{maximize}_\theta E_{s\rho_{\theta_{old}}, a\pi_{\theta_{old}}}[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} A_{\theta_{old}}(s,a)]$$

subject to

$$E_{s\rho_{\theta_{old}}}[D_{KL}(\pi_{\theta_{old}}(\cdot|s)||\pi_\theta(\cdot|s))] \leq \delta$$

This ends the theoratical part of TRPO. For application, we get the sampling data, get the mean and solve the optimization problem.

5.2 Proximal Policy Optimization

Proximal Policy Optimization (PPO) methods [Schulman et al., 2017b] are much simpler than TRPO yet achieve the performance comparable to TRPO. Let $r_t(\theta)$ denote the ratio of the probability under the new policy to that under the old policy

$$r_t(\theta) = \frac{\pi_\theta(a_t, s_t)}{\pi_{\theta_{\text{old}}}(a_t, s_t)}.$$

Then the objective of TRPO can be rewritten as

$$J(\theta) = \mathbb{E}\left[r_t(\theta) A_{\theta_{\text{old}}}\right],$$

subject to the KL-distance constraint. Without this constraint, maximizing $J(\theta)$ can lead to large policy updates and makes the algorithm unstable. PPO deals with the issue by clipping extreme probability ratios, forcing the ratio to stay in the ε neighborhood of 1, namely $(1-\varepsilon, 1+\varepsilon)$. The PPO objective is

$$J_{\text{clipped}}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta)A_{\theta_{\text{old}}}, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)A_{\theta_{\text{old}}}\right)\right]$$

Note that we take the min of the original objective and clipped objective, making the new objective a lower bound of the original objective. Thus, the update is more conservative.

Further algorithms: PPO2, and Distributed PPO (DPPO).

6 Additional Algorithms

6.1 A3C

Besides focusing on a single actor, multiple actors are common for distributed systems. Asynchronous methods are proposed for this setting [Mnih et al., 2016]. In Asynchronous Advantage Actor-Critic (A3C), actors talk to global parameters independently, so they might have different policies. Advantage Actor-Critic (A2C) instead aggregate these actors' data and merge to a same policy.

6.2 DPG

Deterministic policy gradient (DPG) [Silver et al.] models the policy as a deterministic decision. $a = \mu(s)$.

$\rho_0(s)$: initial distribution over states

$\rho^\mu(s \to x, k)$: same definition as in explaining TRPO

$\rho^\mu(s')$: Discounted state distribution $\int_s \sum_0^\infty \gamma^{k-1}\rho_0(s)\rho^\mu(s \to s'.k)ds$

In this case, the reward function becomes: $J(\theta) = \int_S \rho^\mu(s)Q(s, \mu_\theta(s))ds$ This also yields the Deterministic policy gradient theorem. Similar to the Policy Gradient Theorem, we can calculate the derivative

$$\begin{aligned}
&\nabla_\theta J(\theta) \\
=&\int_S \rho^\mu(s)\nabla_a Q^\mu(s,a)\nabla_\theta \mu_\theta(s)|_{a=\mu_\theta(s)}ds \\
=&E_{s\sim\rho^\mu}[\nabla_a Q^\mu(s,a)\nabla_\theta\mu_\theta(s)|_{a=\mu_\theta(s)}]
\end{aligned}$$

The following procedure is similar to our previous algorithms.

Other algorithms:

Deep Deterministic policy gradient (DDPG) [Lillicrap et al., 2019] combines DQN and DPG.

Distributed Distributional DDPG (D4PG) [Barth-Maron et al., 2018] improves DDPG to make it run in distributional cases.

7 Summary

There are various kinds of policy gradient methods for different purposes. In this paper, we only discuss the core theory with an analysis emphasis. Policy gradient methods are complicated and many are related to deep learning methods, making the theoretical analysis more challenging than traditional statistical learning.

References

Policy Gradient Algorithms. https://lilianweng.github.io/2018/04/08/policy-gradient-algorithms.html, Apr. 2018.

G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. Tb, A. Muldal, N. Heess, and T. Lillicrap. Distributed Distributional Deterministic Policy Gradients. In *International Conference on Learning Representations*, Feb. 2018.

K. Cobbe, J. Hilton, O. Klimov, and J. Schulman. Phasic Policy Gradient. *arXiv:2009.04416 [cs, stat]*, Sept. 2020.

T. Degris, M. White, and R. S. Sutton. Off-Policy Actor-Critic. *arXiv:1205.4839 [cs]*, June 2013.

S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pages 267–274, 2002.

V. R. Konda and J. N. Tsitsiklis. Actor-Critic Algorithms.

V. R. Konda and J. N. Tsitsiklis. OnActor-Critic Algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, Jan. 2003. ISSN 0363-0129, 1095-7138. doi: 10.1137/S0363012901385691.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv:1509.02971 [cs, stat]*, July 2019.

A. R. Mahmood. Weighted importance sampling for off-policy learning with linear function approximation.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *arXiv:1602.01783 [cs]*, June 2016.

J. Peters and S. Schaal. Natural Actor-Critic. *Neurocomputing*, 71(7):1180–1190, Mar. 2008. ISSN 0925-2312. doi: 10.1016/j.neucom.2007.11.026.

J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust Region Policy Optimization. *arXiv:1502.05477 [cs]*, Apr. 2017a.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *arXiv:1707.06347 [cs]*, Aug. 2017b.

J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *arXiv:1506.02438 [cs]*, Oct. 2018.

D. Silver. Lecture 7: Policy Gradient.

D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic Policy Gradient Algorithms.

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

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation.

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