Reinforcement learning: variance reduction via martingale representation

Vladimir Dmitriev

HDI HSE

September 3, 2021

Reinforcement learning: Setup

- Agent interacts with the system in discrete time steps
- In each step agent produce some action, which influence both agent and environment
- Each step agent receive "reward" - custom value of current "decision"
- Goal of the learning algorithm is to find good strategy in terms of cumulative reward

... formal description ahead!

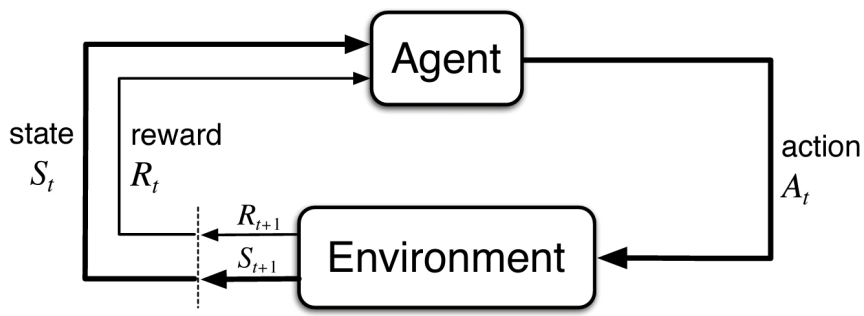


Figure: Scheme of agent-environment interactions [R. Sutton et al. 2000]

Reinforcement learning: Practice & Applications

- Robotics - optimal control tasks [Argall et al. 2009]
- HPC optimal resources allocation [Mao et al. 2016]
- City traffic control - optimization of traffic light regimes etc [Arel et al. 2010]
- Games simulators (Chess & Go) - now achieve superhuman level [Silver et al. 2018]
- Online advertising [Cai et al. 2017]
- Research on cognition and human decision-making [Lee and Seo 2015]

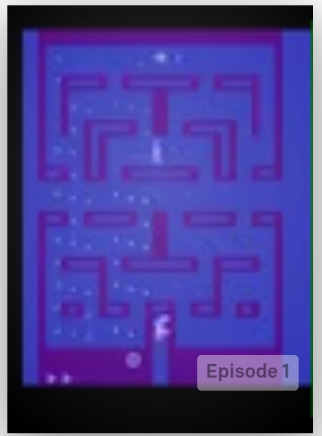
Benchmarks for algorithm development https://gym.openai.com/

Reinforcement learning: Benchmarks

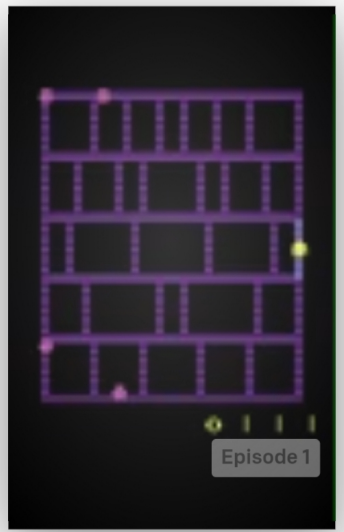
FetchPickAndPlace-v1
Lift a block into the air.

FetchPush-v1
Push a block to a goal position.

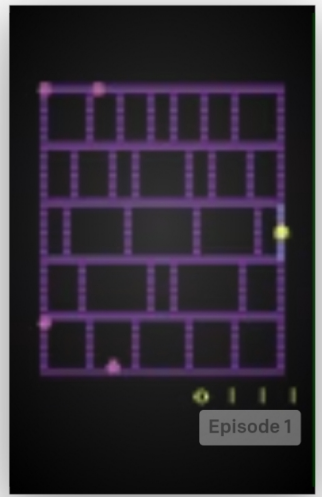
FetchReach-v1
Move Fetch to a goal position.



Alien-v0
Maximize score in the game Alien, with screen images as input



Amidar-ram-v0
Maximize score in the game Amidar, with RAM as input



Amidar-v0
Maximize score in the game Amidar, with screen images as input

Markov Decision Process

- Consider S - state space, $(S_t)_{t>0}$ - sequence of states which agent traverses.
- Let A - action space, $(A_t)_{t>0}$ - sequence of random actions, which are attributed to the agent
- Markovian transition kernel $p_t(s'|s,a) = P_t(S_{t+1} = s'|S_t = s, A_t = a)$ probability of transition to state s' having current state-action pair (s,a)
- Reward for each step - could be deterministic : $R : \mathrm{S} \times \mathrm{S} \times \mathrm{A} \to \mathbb{R}$ or probabilistic $R_t \sim p_r(\cdot|s_{t+1}, s_t, a_t)$;
- At each step t agent has information about environment current state s_t and about history of its interactions $h_t = (s_1, a_1, ..., s_{t-1}, a_{t-1})$. One epoch of decision making gives agent current action a_t.
- Having (s_t, a_t) environment change its state on s_{t+1}, which is sampled from the transition kernel. Agent receive one step reward R_t
- We would consider common case of deterministic rewards and time-homogeneous transition kernels

Markov Decision Process: Policy

- Policy - decision making procedure, could be deterministic

$$a_t = \pi_t(h_{t-1}, s_t)$$

  or probabilistic $a_t \sim \pi_t(\cdot|h_{t-1}, s_t)$
- If policy depends only on s_t it is called Markovian. We would consider only time-homogeneous Markovian policies.
- Note that for Markovian policy state sequence is Markovian:

$$p(s'|s) = \sum_{a \in A_s} \pi(a|s) p(s'|s, a)$$

  and arbitrary path distribution could be computed:

$$P(S_t = s_t, A_t = a_t, S_{t+1} = s_{t+1} ..., S_T = s_T, A_T = a_T | h_{t-1}) =$$

$$= \prod_{k=t}^{k=T} \pi(A_k = a_k | S_k = s_k) p(S_k = s_k | S_{k-1} = s_{k-1}, A_{k-1} = a_{k-1})$$

Value functions

Definitions [R. S. Sutton and Barto 2018]

Let's fix some policy π. The time-homogeneous value function is

$$V_\pi(s) = \mathbb{E}[\sum_{k=0}^{T} \gamma^k R_{k+1} | S = s]$$

The time-homogeneous action-value function is

$$Q_\pi(s, a) = \mathbb{E}[\sum_{k=0}^{T} \gamma^k R_{k+1} | S = s, A = a]$$

where T could stand for infinity, finite horizon or random stopping time

Policy π' is said to be "better" then policy π if

$$V_{\pi'}(s) \geq V_\pi(s) \; \forall s \in S$$

It is denoted as $\pi' \geq \pi$
Policy $\pi^\star$ is said to be optimal (in family Π) if

$$\pi^\star \geq \pi \; \forall \pi \in \Pi$$

Bellman equation

Consider case of finite or infinite horizon T. Let policy and transition kernel be time-homogeneous.Let R_t be reward on step t.

$$V_\pi(s_t) = \mathbb{E}[\sum_{k=t}^{T} \gamma^{k-t} R_k | S_t = s_t] =$$

$$= \mathbb{E}[R_t | S_t = s_t] + \gamma \mathbb{E}[\sum_{k=t+1}^{T} \gamma^{k-(t+1)} R_k | S_t = s_t] =$$

$$= \mathbb{E}[R_t | S_t = s_t] + \gamma \mathbb{E}[V_\pi(s_{t+1})]$$

The same expansion could be written for state-action value function

$$Q_\pi(s_t, a_t) = \mathbb{E}[\sum_{k=t}^{T} \gamma^{k-t} R_k | S_t = s_t, A_t = a_t] =$$

$$= \mathbb{E}[R_t | S_t = s_t, A_t = a_t] + \gamma \mathbb{E}[Q_\pi(s_{t+1}, a_{t+1})]$$

Bellman equation

For important case of deterministic rewards $R : S \times A \to D \subset \mathbb{R}$ and discrete state and action spaces:

$$V_\pi(s) = \sum_{a \in A_s} R(s,a)\pi(a|s) + \gamma \sum_{s' \in S, a \in A_s} V_\pi(s')p(s'|s,a)\pi(a|s)$$

$$Q_\pi(s) = R(s,a) + \gamma \sum_{s' \in S, a' \in A_{s'}} Q_\pi(s',a')p(s'|s,a)\pi(a'|s')$$

- It gives set of linear equations for value function, which could be explicitly solved in finite case
- In Reinforcement learning setup transition kernel isn't provided.
- Problem of RL is how to compute value function and found optimal policies or suboptimal policies with good performance.

Optimal policy

- Optimal policy might not exist.
- If $|A| < \infty$ then optimal policy exists.

Bellman optimality criterion

Policy π is optimal iff $\forall s \in S \;\; \pi(a|s) > 0 \Leftrightarrow a \in Argmax_{a \in A_s}[Q_\pi(s, a)]$

For optimal value function Bellman optimality equations hold(again, suppose that rewards are deterministic):

$$V^\star(s) = max_{a \in A_s}[R(s, a) + \gamma \mathbb{E}[V^\star(s')]]$$

$$Q^\star(s, a) = R(s, a) + \gamma \mathbb{E}[max_{a \in A_s} Q(s', a')]$$

Classical algorithms for policy improvement are based on optimality equation

Dynamic programming

Introduce Bellman control operator $\mathbb{B}[Q]$ on space of Q-functions with norm $||f(x)|| = sup_{x \in D}|f(x)|$

$$\mathbb{B}[Q_\pi(s,a)] = R(s,a) + \gamma \mathbb{E}[max_{a \in A_s} Q_\pi(s',a')]$$

Then the following theorem holds:

Contraction theorem for Bellman control operator

For $\gamma < 1$ Bellman operator is contraction and then has one stationary point.

$$\mathbb{B}[Q(s,a)] = Q(s,a) \Leftrightarrow Q = Q^\star$$

- Bound for error of this algorithm is available [Singh and Yee 1996]
- Inapplicable for RL setup since transition kernel is demanded
- Even for known kernel in case of large state space algorithm becomes inefficient

Example I

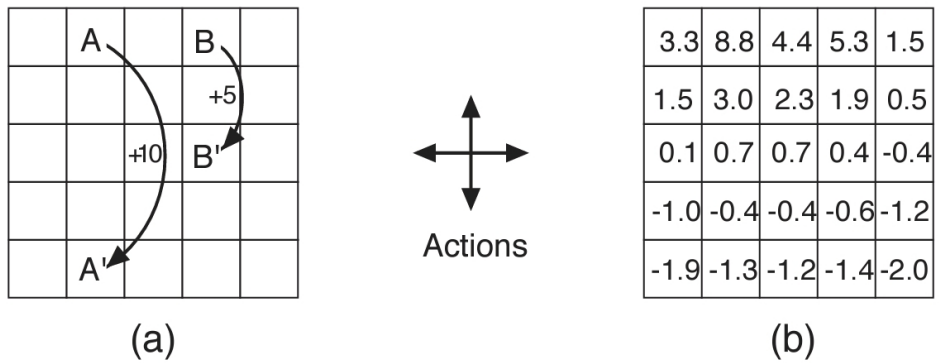


Figure 3.5: Grid example: (a) exceptional reward dynamics; (b) state-value function for the equiprobable random policy.

Figure: R. S. Sutton and Barto 2018

Example II

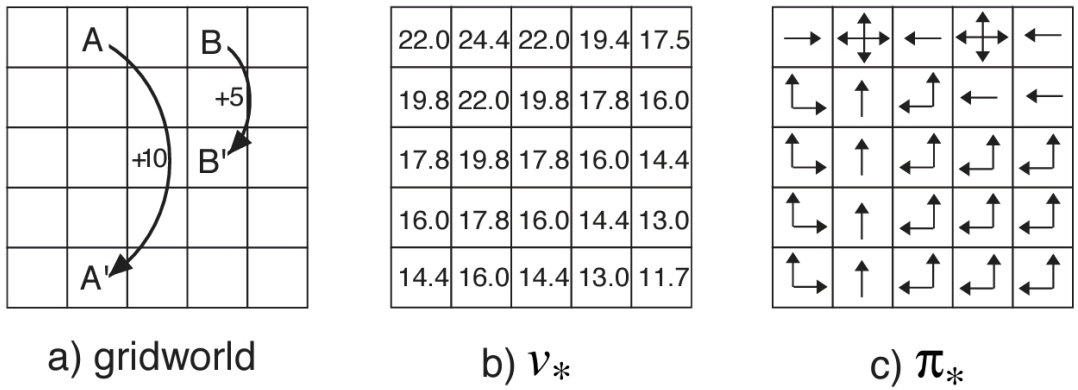


Figure 3.8: Optimal solutions to the gridworld example.

Figure: R. S. Sutton and Barto 2018

Drawbacks of MDP techniques and possible alternatives

- Most classical MC and TD techniques with proven convergence applicable only for tabular case , i.e. finite state and action spaces
- Even finite, but large state space makes tabular algorithms inapplicable
- Instead of tabular setup, one could consider approximations for Q-functions.
- It could be linear approximation (moderate possible to prove convergence, Robbins-Monro procedure), or ANN (no results on convergence, state-of-the-art performance on real world tasks) Sigaud and Garcia 2013

Policy Gradient I

In what follows we describe policy gradient methods. This methods are used extensively in applications, since they were proposed by Williams 1992, however there is no complete theoretical analysis of their convergence.
Consider parametric family of conditional distributions on action space. We consider distributions with densities or distributions on discrete sets.

Example I

Let $S = D \in \mathbb{R}^n$, $A = \mathbb{R}$ π_θ could be taken in form:

$$\pi_\theta(a|s) = \frac{1}{\sqrt{2\pi\sigma(s)^2}} exp(-\frac{(a-\mu(s))^2}{2\sigma^2(s)})$$

where $\mu : S \to D_2 \subseteq \mathbb{R}$, $\sigma : \sigma : S \to D_2 \subseteq \mathbb{R}^+$ This normal policy is often used in continuous control. It could be easily generalized for multidimensional actions by setting independent distribution for each component. Lillicrap et al. 2015 for examples

Policy Gradient II

Example II

Let $S = \Omega_S$, $A = \Omega_A = \{\omega_1^a, \omega_2^a, ..., \omega_k^a\}$, then π_θ could be taken in form:

$$\pi_\theta(a_i|s) = \frac{exp(-\beta_\theta^i(s))}{\sum_{j=1}^k exp(-\beta_\theta^j(s)} \; \forall i \in \{1, ..., k\}$$

where $\beta^i : \Omega_S \to D \subseteq \mathbb{R}$
This policy is called Softmax policy. Set Ω_S could be of arbitrary nature.
This policies are often used for Atari setups [R. Sutton et al. 2000]

Policy Gradient III

Here we follow open source materials of Sergey Levine course on RL
After fixing a distribution value function becomes function of parameters θ. Distribution over finite trajectories $\tau = \{s_1, a_1, s_2, a_2, ..., s_T, a_T\}$

$$p_\theta(\tau) = p_0(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

In that terms optimal θ wrt criterion is

$$\theta^\star = argmax\ \mathbb{E}_\tau[\sum_{t=1}^{T} \gamma^{t-1} R_t]$$

Denote

$$R(\tau) = \sum_{t=1}^{T} \gamma^{t-1} R_t$$

In what follows we suppose that $T = \infty$ and parameter values θ don't affect supp of policy distribution.

Policy Gradient IV

Consider target functional:

$$J(\theta) = \int_{\Omega} p(\tau) R(\tau) d\tau$$

We would construct stochastic gradient procedure. Compute gradient :

$$\nabla_\theta J(\theta) = \int_{\Omega} \nabla_\theta [p_\theta(\tau)] R(\tau) d\tau$$

For that, use log-derivative trick:

$$\nabla_\theta p_\theta(\tau) = \frac{\nabla_\theta p_\theta(\tau)}{p_\theta(\tau)} p_\theta(\tau) =$$

$$= p_\theta(\tau) \nabla_\theta log[p_\theta(\tau)]$$

Policy Gradient V

Combining log-derivative with previous we achieve

$$\nabla_\theta J(\theta) = \int_\Omega p_\theta(\tau) \nabla_\theta log[p_\theta(\tau)] R(\tau) d\tau$$

Look at gradient:

$$logp_\theta(\tau) = log(p(s_1)) + \sum_{t=1}^{T} log\pi_\theta(a_t|a_t) + \sum_{t=1}^{T} log[p(s_{t+1}|s_t, a_t)]$$

Then

$$\nabla_\theta log[p_\theta(\tau)] = \nabla_\theta \sum_{t=1}^{T} log\pi_\theta(a_t|a_t)$$

Policy Gradient VI

Finally

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau[(\sum_{t'=1}^{T} \gamma^{t'-1} R_{t'}) \sum_{t=1}^{T} \nabla_\theta log \pi_\theta(a_t|s_t)]$$

- We just prove policy gradient theorem Williams 1992
- Theory of stochastic gradient Schulman et al. 2016 allows to use sample estimate of the gradient.

Policy Gradient VII

Assume again, that we sample N trajectories:

$$\nabla_\theta \hat{J}(\theta) = \frac{1}{N} \sum_{i=1}^{N} [(\sum_{t'=1}^{T} \gamma^{t'-1} R_{t'}^i) \sum_{t=1}^{T} \nabla_\theta log \pi_\theta(a_t^i | s_t^i)]$$

It also has been shown [Williams 1992] that

$$\nabla_\theta \hat{J}(\theta) = \frac{1}{N} \sum_{i=1}^{N} [\sum_{t=1}^{T} (\sum_{t'=t}^{T} \gamma^{t'-t} R_{t'}^i) \nabla_\theta log \pi_\theta(a_t^i | s_t^i)]$$

Stochastic gradient ascent for policy optimization:

$$\theta_{q+1} \leftarrow \theta_q + \alpha_q \nabla_\theta \hat{J}(\theta_q)$$

Control variates

- Let X be sample of some parametric distribution.
- Assume that $\hat{s}(X)$ - unbiased estimate of some distribution parameter.

$$\mathbb{E}[\hat{s}] = s$$

- Let $cv(X)$ be sample function, and

$$\mathbb{E}[cv(X)] = 0$$

- We can consider new estimate $\hat{s}' = \hat{s} - cv$.
- Variance would be reduced if

$$\mathbb{V}[\hat{s} - cv] < \mathbb{V}[\hat{s}]$$

- Since

$$\mathbb{V}[\hat{s} - cv] = \mathbb{V}[\hat{s}] + \mathbb{V}[cv] + 2Cov(cv, \hat{s})$$

control variate will be benefitial if

$$-\mathbb{V}[cv] \geq 2Cov(cv, \hat{s})$$

Control variates in Policy Gradient I

- Denote

$$R(\tau_t^i) = \sum_{t'=t}^{T} \gamma^{t'-t} R_{t'}^i$$

- First attempt to reduce variance in policy gradient estimate was to introduce constant baseline b:

$$\nabla_\theta \hat{J}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} (R(\tau_t^i) - b) \nabla_\theta log \pi_\theta(a_t^i | s_t^i)$$

- Optimization of variance gives the following result [Weaver and Tao 2013]:

$$b = \frac{\mathbb{E}\left[||\nabla_\theta \pi_\theta(A|S)||^2 R(\tau_i)\right]}{\mathbb{E}\left[||\nabla_\theta \pi_\theta(A|S)||^2\right]}$$

Control variates in Policy Gradient II

- Next attempt was to use function of state since it doesn't introduce bias. For arbitrary $b(s)$:

$$\int_{\Omega} b(s)\pi_\theta(a|s)\nabla_\theta log(\pi_\theta(a|s))da =$$

$$= \int_{\Omega} b(s)\nabla_\theta\pi_\theta(a|s)da = \nabla_\theta \mathbb{E} b(s) = 0$$

- It have been shown that taking estimate of $V(s)$ [Sigaud and Garcia 2013] gives good results and decrease variance. Corresponding algorithm is known as REINFORCE with baseline

Policy gradient methods maximize the expected total reward by repeatedly estimating the gradient $g := \nabla_\theta \mathbb{E}\left[\sum_{t=0}^{\infty} r_t\right]$. There are several different related expressions for the policy gradient, which have the form

$$g = \mathbb{E}\left[\sum_{t=0}^{\infty} \Psi_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right], \tag{1}$$

where Ψ_t may be one of the following:

1. $\sum_{t=0}^{\infty} r_t$: total reward of the trajectory.
2. $\sum_{t'=t}^{\infty} r_{t'}$: reward following action a_t.
3. $\sum_{t'=t}^{\infty} r_{t'} - b(s_t)$: baselined version of previous formula.
4. $Q^\pi(s_t, a_t)$: state-action value function.
5. $A^\pi(s_t, a_t)$: advantage function.
6. $r_t + V^\pi(s_{t+1}) - V^\pi(s_t)$: TD residual.

The latter formulas use the definitions

$$V^\pi(s_t) := \mathbb{E}_{\substack{s_{t+1:\infty}, \\ a_{t:\infty}}}\left[\sum_{l=0}^{\infty} r_{t+l}\right] \qquad Q^\pi(s_t, a_t) := \mathbb{E}_{\substack{s_{t+1:\infty}, \\ a_{t+1:\infty}}}\left[\sum_{l=0}^{\infty} r_{t+l}\right] \tag{2}$$

$$A^\pi(s_t, a_t) := Q^\pi(s_t, a_t) - V^\pi(s_t), \quad \text{(Advantage function)}. \tag{3}$$

Figure: Schulman et al. 2016

Theory and Algorithm

- Theory based on Belomestny et al. 2019
- Consider Markov process of a form

$$X_{i+1} = \Phi(X_i, \xi_{i+1}) \quad i = 0, 1, \ldots \; X_0 = x$$

$$\forall i \; X_i \in \mathbb{X}$$

- $\xi_i \in \mathbb{R}^m$ - i.i.d distributed with P_ξ
- Φ Borel-measurable functions:

$$\Phi : \mathbb{X} \times \mathbb{R}^m \to \mathbb{X}$$

- This setup can be exploited for many continuous control tasks

Theory and Algorithm

Let $(\phi_k)_{k\geq 0}$ be a complete orthonormal system in $L^2(\mathbb{R}^m, P_\xi)$ with $\phi_0 \equiv 1$. In particular,

$$\mathsf{E}[\phi_i(\xi)\phi_j(\xi)] = \delta_{ij}, \quad i,j \in \mathbb{Z}_+.$$

Observation

The random variables $\phi_k(\xi)$, $k \geq 1$, are centered, i.e. $\mathbb{E}[\phi_k(\xi)] = 0$

Example

For r.v. ξ with normal distribution P_ξ we can take multivariate Hermite polynomials.

Theory and Algorithm

Theorem

Let $\{\phi_k\}_{k\geq 0}$ be complete orthonormal system in $\mathbb{L}_2(P_\xi)$. Then for any bounded function f the following representation holds:

$$f(X_q) = \mathbb{E}\left[f(X_q)\right] + \sum_{k=1}^{\infty}\sum_{l=1}^{q} a_{q-l,k}(X_{l-1})\phi_k\left(\xi_l\right)$$

where for all $y \in \mathbb{R}^d$ $(\mathbb{X})$,

$$a_{r,k}(y) = \mathbb{E}\left[f(X_r)\phi_k\left(\xi_1\right)|X_0 = y\right] \quad r,k \in \mathbb{N}$$

Observation

$\mathbb{E}\phi_k(\xi_1) = 0 \Rightarrow \sum_{k=1}^{K} a_{r,k}(x)\phi_k(\xi)$ is a valid control variate for $f(X_q)$.

Theory and Algorithm

Corollary

$\{\xi_i\}$ are sampled from standard normal distribution, Hermite polynomials can be taken as orthonormal system. Then the quantity

$$M_{K,r}^q = \sum_{k=1}^{K} \sum_{r=0}^{q-1} a_{r,k}(X_{q-r-1}) H_k(\xi_{q-r})$$

for the fixed $K > 0$ has zero mean (i.e. $\mathbb{E}[f(X_q)] = \mathbb{E}[f(X_q) - M_{K,r}^q]$ and $M_{K,r}^q$ can be viewed as a control variate.

Theory and Algorithms

Remark 1

Variance of the new estimate could be calculated:

$$Var[f(X_q) - M^q_{K,r}] = \sum_{k=K+1}^{\infty} \sum_{r=0}^{q-1} \mathbb{E}(a^2_{rk}(X_{q-r-1}))$$

If coefficients $a_{r,k}$ decays fast enough with $k \to \infty$, variance reduction can be done with this control variate

Remark 2

Note that the coefficients $a_{r,k}$ can be alternatively expressed as

$$a_{r,k}(x) = \mathbb{E}\left[\phi_k(\xi)\, Q_r(\Phi(x,\xi))\right] \quad \text{with } Q_r(x) = \mathbb{E}\left[f(X_r)|X_0 = x\right],\ r \in \mathbb{N}.$$

It may be more convenient to work with Q_r directly.

Note that for ergodic Markov chain $(X_k)_{k\geq 0}$, $Q_r(x)$ converges to $\pi(f)$ exponentially fast, hence $a_{r,k}(x)$ goes to zero.

Theory and Algorithms

- One way to reconstruct coefficients Q_r is regression on trajectories:

$$\hat{Q}_r(x) = \sum_{b=1}^{B} \beta_b^{(r)} \phi_b(x)$$

where coefficients are found from solving least squares problem:

$$\beta^{(r)} = \mathit{argmin} \sum_{s=0}^{n-r} |f(X_{r+s}) - \sum_{b=1}^{B} \beta_b^{(r)} \phi_b(X_s)|^2$$

- In case of regression we need to calculate integral:

$$a_{r,k}(x) = \sum_{b=1}^{B} \beta_b^{(r)} \int_{\Omega} \phi_k(\xi)\phi_b(\xi) P_\xi(d\xi)$$

- One another powerful option - approximate $a_{r,k}(x)$ directly with ANNs.

Theory and Algorithms

Algorithm 2: General Policy Gradient algorithm with buffer replay

initialization - init buffer replay R, init parametric critic $Q_w(a, s)$ or $V_w(s)$, init parametric policy $\pi_\theta(a|s)$, N - iterations number, $p(s_0)$ - distribution for starting states, E - number of trajectories to play for one iteration;

while *num_it* $< N$ **do**

  num_it+=1;

  for $e = 0;\ e < E;\ e = e + 1$ **do**

    $s_{e,0} \sim p(s_0)$;

    while $s_{t,e} = s_{terminal}$ **do**

      $a_{t,e} \sim \pi(\cdot|s_{t,e})$;

      $s_{t+1,e} \sim p(\cdot|s_{t-1,e}, a_{t-1,e})$;

      $r_{t,e} = r(s_{t,e}, a_{t,e})$;

      collect step to buffer $\mathcal{B}$;

    end

    add $\mathcal{B}$ *in* $\mathcal{R}$;

  end

  Take $\bar{N}$ gradient steps on Q_w (or V_w) using samples from $\mathcal{R}$;

  Compute coefficients* Ψ_t ;

  Compute correction* to the gradient g_t ;

  Compute gradient:

$$\nabla J = \sum_{t=0}^{T} (\Psi_t \nabla_\theta [log(\pi_\theta(a_t|s_t))] + g_t)$$

  Update θ with gradient ∇J;

$$\theta_{new} = \theta_{old} + \alpha \nabla J$$

end

Theory and Algorithms

Algorithm 6: Add control variate to Ψ_t

input - buffer $\mathcal{B}$, Ψ_t, $a_{r,k'}$ - set of ANN - approximators for control variate, k, l - parameters for number of polynomials and lag respectively. γ - discount factor; set $\mu = \frac{1}{length(\mathcal{B})} \sum_{t=1}^{length(\mathcal{B})} \Psi_t$;

for $t = length(\mathcal{B}) - 1;\ t \geq 0;\ t = t - 1$ **do**

  for $k' = 1;\ k' \leq k;\ k' = k' + 1$ **do**

    for $r = 1;\ r \leq l;\ r = r + 1$ **do**

      $\Psi'_t = \Psi_t + a_{r,k'}(S_t, A_t)\phi_{k'}(\xi_t)$;

    end

  end

end

Compute loss for control variate ;

$$L = \frac{1}{length(\mathcal{B})}\Big(\sum_{t=0}^{length(\mathcal{B})} (\Psi'_t - \mu)^2 + \lambda \sum_{k',r,t} a^2_{r,k'}(S_t, A_t)\Big)$$

Update weights of $a_{r,k'}$ minimizing L;

Frozen Lake environment, REINFORCE

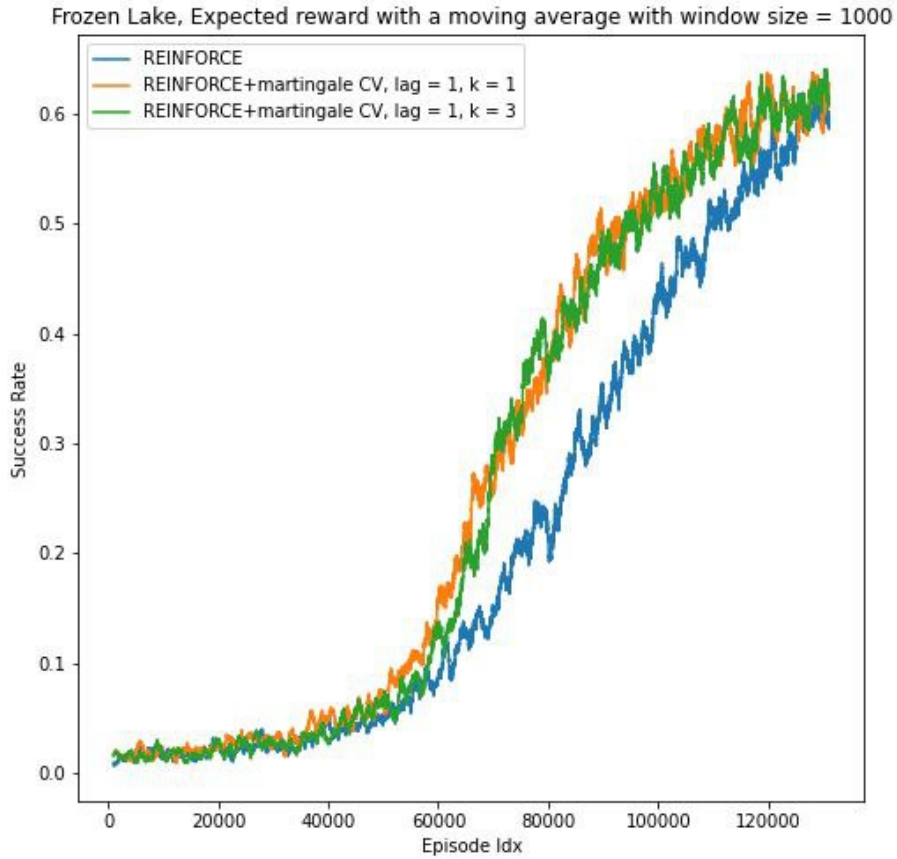


Figure: Policy evaluation during learning of the REINFORCE algorithm for FrozenLake environment. Blue plot is REINFORCE without martingale CV, other plots demonstrate the increase of improvements with adding more polynomials to martingale CV

CartPole, A2C

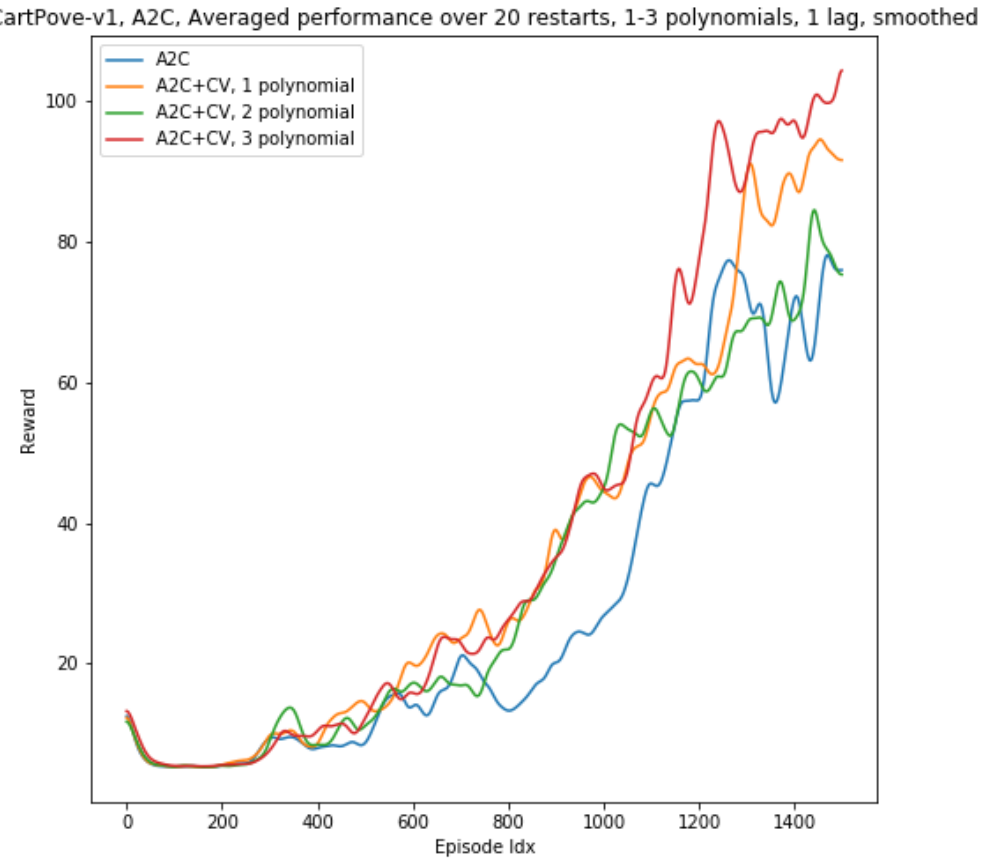


Figure: Policy evaluation during learning of the A2C algorithm for CartPole-v1 environment. Blue plot is A2C without martingale CV, other plots demonstrate the increase of improvements with adding more polynomials to martingale CV

Regression test

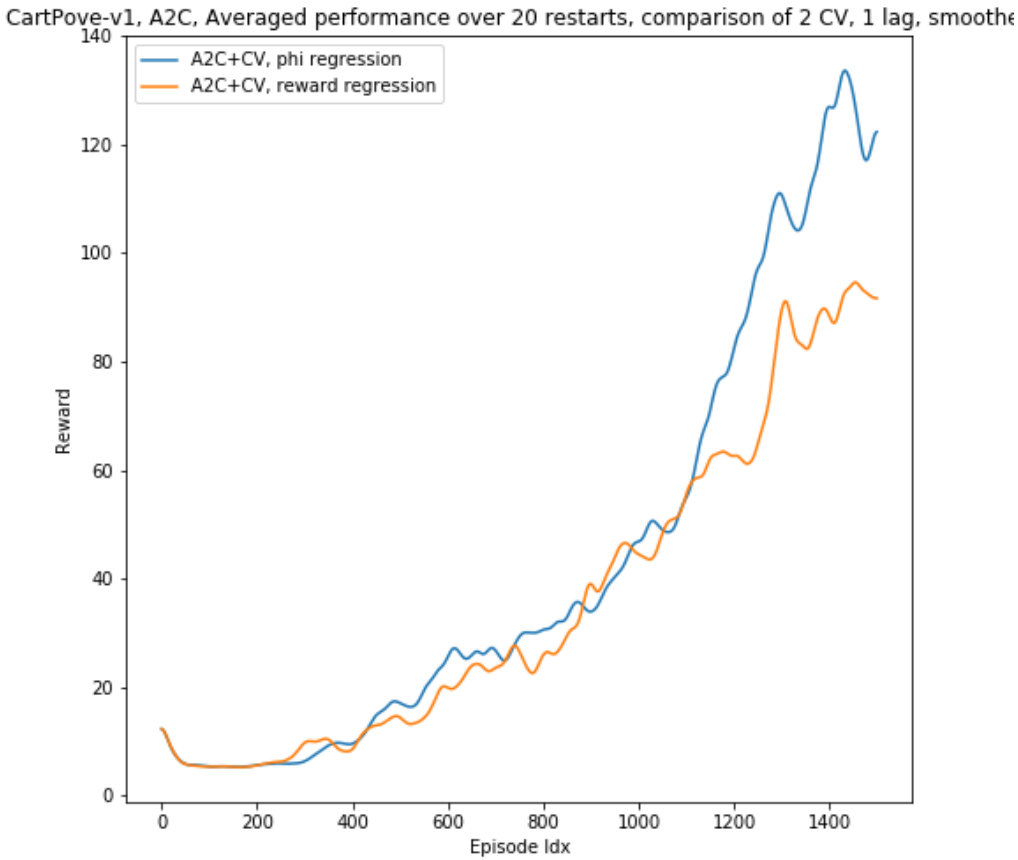


Figure: Policy evaluation during learning of the A2C algorithm for CartPole-v1 environment. Blue plot is A2C with martingale CV with $\Psi_t = A_t$, orange plot - $\Psi_t = r_t$

Martingale CV vs GAE

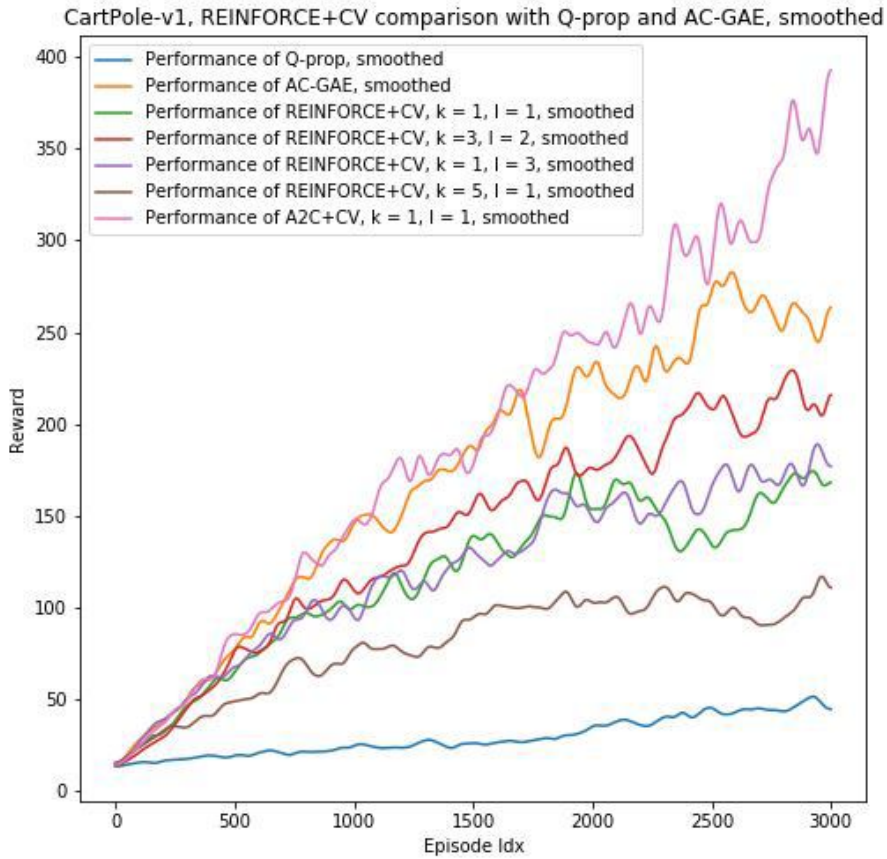


Figure: Comparison of Q-prop and AC-GAE with REINFORCE+CV

Appendix: Theory and Algorithms: Basic Idea

- Theory based on Belomestny et al. 2019
- Consider Markov process of a form

$$X_{i+1} = \Phi(X_i, \xi_{i+1}) \quad i = 0, 1, \ldots \; X_0 = x$$

$$\forall i \; X_i \in \mathbb{X}$$

where $\xi_i \in \mathbb{R}^m$ - i.i.d distributed with P_ξ, Φ Borel-measurable functions. In RL case $X_i = [S_i, A_i]$

- Target function $f(X)$ has decomposition:

$$f(X_q) = \mathbb{E}\left[f(X_q)\right] + \sum_{k=1}^{\infty} \sum_{l=1}^{q} a_{q-l,k}(X_{l-1}) \phi_k\left(\xi_l\right)$$

where $\phi_k k \geq 0$ be a complete orthonormal system in $L^2(\mathbb{R}^m, P_\xi)$ and $y \in \mathbb{R}^d$ $(\mathbb{X})$,

$$a_{r,k}(y) = \mathbb{E}\left[f(X_r)\phi_k\left(\xi_1\right) | X_0 = y\right] \quad r, k \in \mathbb{N}$$

- Arbitrary number of terms in right hand side sum could be taken for control variate. ANN would be used to approximate $a_{r,k}$

Appendix: Policy Gradient: Example

Normal Policy

Consider normal policy with fixed covariance matrix

$$\pi_\theta(a|s) \sim \exp(-\frac{||a-\mu_\theta(s)||^2_{\Sigma^{-1}}}{2})$$

Then

$$log[\pi_\theta(a|s)] = const + -\frac{||a-\mu_\theta(s)||^2_{\Sigma^{-1}}}{2}$$

$$\nabla_\theta \hat{J}(\theta) = \frac{1}{2}\Sigma^{-1}(a-\mu_\theta(s))\frac{d\mu_\theta(s)}{d\theta}$$

Last derivative could be computed by backpropagation in case of ANN approximation

Stochastic gradient estimates suffer from high variance. Variance reduction techniques are applicable to modify gradient estimate to reduce its variance

Policy Gradient II

Example II

Let $S = \Omega_S$, $A = \Omega_A = \{\omega_1^a, \omega_2^a, ..., \omega_k^a\}$, then π_θ could be taken in form:

$$\pi_\theta(a_i|s) = \frac{exp(-\beta_\theta^i(s))}{\sum_{j=1}^k exp(-\beta_\theta^j(s)} \; \forall i \in \{1, ..., k\}$$

where $\beta^i : \Omega_S \to D \subseteq \mathbb{R}$
This policy is called Softmax policy. Set Ω_S could be of arbitrary nature. This policies are often used for Atari setups [R. Sutton et al. 2000]

Example III

In case when action space is essentially bounded in $\mathbb{R}$, Kumaraswamy distribution could be used: Let $S = \Omega_S$, $A = (0, 1)$ then π_θ could be taken in as:

$$\pi_\theta(a|s) = \alpha_\theta(s)\beta_\theta(s)a^{\alpha_\theta(s)-1}(1 - a^{\alpha_\theta(s)})^{\beta_\theta(s)-1}$$

where $\alpha, \beta : \Omega_S \to \mathbb{R}^+$

Appendix: Monte Carlo and Temporal Difference

- Monte Carlo estimates of Q-function and value function could be used under mild conditions on the Markov chain
- Assume finite MDP, agent experience is saved as trajectories $Tr_i = \{\langle s_t^i, a_t^i, r_t^i \rangle\}_{t=1}^{T_i}$
- Let $I = \{i : s_0^i = s\}$, then $\hat{V}(s) = \frac{1}{|I|} \sum_{i \in I} \sum_{t=1}^{T_i} \gamma^{t-1} r_t^i$
- Extensive studies for RL applications, see [R. Sutton et al. 2000] for more references.
- Temporal difference provide iterative updates for estimates of Q-function or value function.
- Convergence is proven for tabular case [R. Sutton et al. 2000]

$$Q^{n+1}(s_t, a_t) \leftarrow Q^n(s_t, a_t) + \alpha_n [R(s_t, a_t) + \gamma Q^n(s_{t+1}, a_{t+1}) - Q^n(s_t, a_t)]$$

Appendix: Markov Decision Process: Horizon and Optimality criterion

There are following options for MDP:

- Finite MDP - set maximal number of steps after which the agent stops
- Stopping time - set random variable which defines when game stops (remember gambler's ruin problem)
- Infinite horizon - number of steps is unbounded (most continuous control tasks)
- The goal of the learning algorithm is to find a such a decision making procedure, which would maximise expected reward for the game with respect to some initial states distribution.
- Among the variety of optimality criterion (see [Puterman 1994]) the standard choice for RL is expected total discounted reward (to be defined)

Markov Decision Process

Reinforcement learning inherits Markov Decision Process formalism. Main features of MDP:

- Consider S - state space, $(S_t)_{t>0}$ - sequence of states which agent traverses.
- Let A - action space, $(A_t)_{t>0}$ - sequence of random actions, which are attributed to the agent

Both state and action spaces could be:

- Finite sets - case of Atari games
- Infinite countable sets - queuing problems & optimal stopping
- Compact subsets of the Euclidean space - case of continuous control tasks with periodic state space (pendulum, double pendulum etc)
- Unbounded subsets of the Euclidean space - general continuous control tasks/
- Borel sets in general Polish spaces (see [Puterman 1994])

Markov Decision Process: Transition kernel

- Markovian transition kernel $p_t(s'|s,a) = P_t(S_{t+1} = s'|S_t = s, A_t = a)$ probability of transition to state s' having current state-action pair (s,a)
- Reward for each step - could be deterministic : $R : \mathrm{S} \times \mathrm{S} \times \mathrm{A} \to \mathbb{R}$ or probabilistic $R_t \sim p_r(\cdot|s_{t+1}, s_t, a_t)$;
- At each step t agent has information about environment current state s_t and about history of its interactions $h_t = (s_1, a_1, ..., s_{t-1}, a_{t-1})$. One epoch of decision making gives agent current action a_t.
- Having (s_t, a_t) environment change its state on s_{t+1}, which is sampled from the transition kernel. Agent receive one step reward R_t
- We would consider common case of deterministic rewards and time-homogeneous transition kernels