SDU

7- Policy Gradient Methods

Melih Kandemir

University of Southern Denmark
Department of Mathematics and Computer Science (IMADA)
kandemir@imada.sdu.dk

Fall 2022

Policy-based RL

- We have thus far attempted to solve the reinforcement learning problems by designing algorithms that update value functions and use them to derive policies.
- There exist an alternative vein in reinforcement learning that suggests modeling the policy distribution straight ahead and fitting its parameters to observations.
- Let us this time choose a policy distribution $\mu_\theta(u|i)$ with parameters θ. Our goal is to minimize the cost-to-go of a state i with respect to policy parameters θ

$$\theta^* = \arg\min_\theta \; J_{\pi_\theta}(i). \tag{1}$$

Advantages of policy-based methods

- Facilitate modeling continuous action spaces.
- Ability to express any action distribution other than $\epsilon-$greedy.
- A small change in the value function parameters could make a previously unlikely action the most likely one. Contrarily, the action probabilities of a policy-based method could be updated proportional to a desired learning rate.
- Usually easier to study the convergence properties of policy-based methods than value-based methods.
- Domain knowledge about a target environment is often available in the form of a policy distribution, such as the behavior patterns of a human driver.

Discriminative versus generative models

Consider the supervised learning problem on a data distribution $(x, y) \sim p(x, y)$ where x are input observations and y are labels. Given a data set $\mathcal{D} = \{(x_i, y_i) | (x_i, y_i) \sim p(x, y), \forall i = 1, \ldots, N\}$

- a **generative model** uses factorization $p(y)p(x|y)$ and learns both: $\arg\max_{\phi,\theta} \frac{1}{N} \sum_{i=1}^{N} p_\phi(y_i) p_\theta(x_i|y_i)$. **Value-based methods** learn $p(\mu)p(J|\mu)$, hence they try to explain the cost generation process.
- a **discriminative model** uses the factorization $p(x)p(y|x)$, integrates out the first one $\mathbb{E}_{p(x)}[p(y|x)] \approx \frac{1}{N} \sum_{i=1}^{N} p(y_i|x_i)$ and learns only the second, i.e. only the label predictor

$$\arg\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} p_\theta(y_i|x_i).$$

Policy-based methods learn $p(\mu|J)$ by doing

$$\mathbb{E}_{J \sim p(J)}[p(\mu|J)] \approx \frac{1}{N} \sum_{m=1}^{N} p(\mu|J_m) \Rightarrow \arg\max_{\theta} \frac{1}{N} \sum_{m=1}^{N} p_\theta(\mu|J_m)$$

where J_m are cost-to-go values observed from MC samples.

The policy-gradient theorem

$$
\begin{aligned}
\nabla_\theta J_\theta(i) &= \nabla_\theta \int \mu_\theta(u|i) Q_\theta(i,u) du \\
&= \int \nabla_\theta \mu_\theta(u|i) Q_\theta(i,u) du + \int \mu_\theta(u|i) \nabla_\theta Q_\theta(i,u) du \\
&= \int \nabla_\theta \mu_\theta(u|i) Q_\theta(i,u) du \\
&+ \int \mu_\theta(u|i) \nabla_\theta \Big\{ \int p^u_{ii'} \Big[g_i + \int \mu_\theta(u'|i') Q_\theta(i',u') du' \Big] di' \Big\} du \\
&= \int \nabla_\theta \mu_\theta(u|i) Q_\theta(i,u) du + \int \mu_\theta(u|i) g_i \Big[\cancelto{0}{\nabla_\theta \int p^u_{ii'} di'} \Big] du \\
&+ \int \mu_\theta(u|i) p^u_{ii'} \nabla_\theta \Big\{ \underbrace{\int \mu_\theta(u'|i') Q_\theta(i',u') du'}_{J_\theta(i')} \Big\} di' du
\end{aligned}
$$

The policy-gradient theorem

$$\nabla_\theta J_\theta(i) = \int \nabla_\theta \mu_\theta(u|i) Q_\theta(i,u) du + \int \mu_\theta(u|i) p^u_{ii'} \nabla_\theta J_\theta(i') di' du$$
$$= \int \nabla_\theta \mu_\theta(u|i) Q_\theta(i,u) du + \mathbb{E}[\nabla_\theta J_\theta(i')]$$

Apply recursion

$$\mathbb{E}[\nabla_\theta J_\theta(i_{k+1})]$$
$$= \mathbb{E}\left[\int \nabla_\theta \mu_\theta(u_{k+1}|i_{k+1}) Q_\theta(i_{k+1}, u_{k+1}) du_{k+1} + \mathbb{E}[\nabla_\theta J_\theta(i_{k+2})] \right]$$
$$= \mathbb{E}\left[\int \nabla_\theta \mu_\theta(u_{k+1}|i_{k+1}) Q_\theta(i_{k+1}, u_{k+1}) du_{k+1} \right] + \mathbb{E}[\nabla_\theta J_\theta(i_{k+2})]$$

and unfold until the end of the episode where $\mathbb{E}[J_\theta(i_N)] = 0$:

$$\nabla_\theta J_\theta(i) = \sum_{k=0}^{N-1} \mathbb{E}_{p(i_k)} \left[\int \nabla_\theta \mu_\theta(u_k|i_k) Q_\theta(i_k, u_k) du_k \right].$$

MC simulation of the policy gradient

- When we are able to build the computational graph of $J_\theta(i)$, we can also compute $\nabla_\theta J_\theta(i)$. But this is possible under two assumptions: i) we have access to the true environment model, ii) the expectations with respect to the environment model are analytically tractable.
- The alternative is MC simulation but how shall we take the inner integral when $\nabla_\theta \pi_\theta(u|i)$ is not a probability distribution from which we can draw samples?

The REINFORCE trick

Theorem

For any distribution $x \sim p(x;\theta)$ parameterized by θ and any function $f(x)$ defined on the support set of $p(\cdot)$, the following identity holds

$$\nabla_\theta \mathbb{E}_{p(x;\theta)}[f(x)] = \int f(x) \nabla_\theta \log p(x;\theta) p(x;\theta) dx. \tag{2}$$

Proof.

$$\begin{aligned}
\nabla_\theta \mathbb{E}_{p(x;\theta)}[f(x)] &= \nabla_\theta \int f(x) p(x;\theta) dx \\
&= \int f(x) \nabla_\theta p(x;\theta) dx \\
&= \int f(x) \frac{\nabla_\theta p(x;\theta)}{p(x;\theta)} p(x;\theta) dx \\
&= \int f(x) \nabla_\theta \log p(x;\theta) p(x;\theta) dx \qquad \square
\end{aligned}$$

Apply the trick to the policy gradient

$$
\begin{aligned}
\nabla_\theta J_\theta(i) &= \sum_{k=0}^{N-1} \mathbb{E}_{p(i_k)} \left[\int \nabla_\theta \mu_\theta(u_k|i_k) Q_\theta(i_k, u_k) du_k \right] \\
&= \sum_{k=0}^{N-1} \mathbb{E}_{p(i_k)} \left[\int \nabla_\theta \mu_\theta(u_k|i_k) \frac{\mu_\theta(u_k|i_k)}{\mu_\theta(u_k|i_k)} Q_\theta(i_k, u_k) du_k \right] \\
&= \sum_{k=0}^{N-1} \mathbb{E}_{p(i_k)} \left[\int \frac{\nabla_\theta \mu_\theta(u_k|i_k)}{\mu_\theta(u_k|i_k)} \mu_\theta(u_k|i_k) Q_\theta(i_k, u_k) du_k \right] \\
&= \sum_{k=0}^{N-1} \mathbb{E}_{p(i_k)} \left[\int \nabla_\theta \log \mu_\theta(u_k|i_k) Q_\theta(i_k, u_k) \mu_\theta(u_k|i_k) du_k \right] \\
&= \sum_{k=0}^{N-1} \mathbb{E}_{p(i_k)\mu_\theta(u_k|i_k)} \left[\nabla_\theta \log \mu_\theta(u_k|i_k) Q_\theta(i_k, u_k) \right]
\end{aligned}
$$

Now do MC simulation

$$\nabla_\theta J_\theta(i) = \sum_{k=0}^{N-1} \mathbb{E}_{p(i_k)\mu_\theta(u_k|i_k)} \Big[Q_\theta(i_k, u_k) \nabla_\theta \log \mu_\theta(u_k|i_k) \Big]$$

For M sampled sequences $\{(i_0^m, u_0^m), (i_1^m, u_1^m), \ldots, (i_{N-1}^m, u_{N-1}^m)\}$

$$Q_\theta(i_k, u_k) \approx \frac{1}{M} \sum_{m=1}^{M} \underbrace{\sum_{j=k}^{N-1} g(i_j^m, u_j^m, u_{j+1}^m)}_{C_k^m} .$$

which gives us a MC estimate of the policy gradient

$$\nabla_\theta J_\theta(i) \approx \frac{1}{M} \sum_{m=1}^{M} \sum_{k=0}^{N-1} C_k^m \nabla_\theta \log \mu_\theta(u_k^m|i_k^m).$$

The corresponding loss is then given as

$$\theta^* := \arg\min_\theta \frac{1}{M} \sum_{m=1}^{M} \sum_{k=0}^{N-1} C_k^m \log \mu_\theta(u_k^m|i_k^m).$$

The REINFORCE algorithm

MC maximization with respect to the policy gradient can be expressed as an algorithm called **REINFORCE**.

repeat ▷ Episodes

act $u_k \sim \mu(u_k|i_k)$ and observe $i_{k+1}, \forall k = 0, \ldots, N$ ▷ Full backup

$C := 0$

for do $k = N - 1 \to 0$

$C := C + g(i_k, u_k, i_{k+1})$

$\theta := \theta - \gamma C \nabla_\theta \log \mu_\theta(u_k|i_k)$

end for

until convergence

Actor-critic methods

- REINFORCE can only update policy network parameters after sampling a full episode, which generates high variance on the gradient estimator.
- One can instead maintain a value predictor alongside the policy function and use it for bootstapping, hence speed up learning and reduce estimator variance. RL algorithms that follow this approach are called **actor-critic** methods.
- The policy function is the **actor**, as the actions are taken based on it, and the cost-to-go estimate is the **critic** as it estimates the consequences of a taken action.

Baselines

Choose an arbitrary function $b(i_k)$, called a **baseline**, and subtract it from the cost-to-go

$$\mathbb{E}\Big[\int \nabla_\theta \mu_\theta(u_k|i_k)\Big[Q_\theta(i_k,u_k) - b(i_k)\Big]du_k\Big]$$

$$= \mathbb{E}\Big[\int \nabla_\theta \mu_\theta(u_k|i_k) Q_\theta(i_k,u_k) du_k\Big] - \mathbb{E}\Big[\int \nabla_\theta \mu_\theta(u_k|i_k) b(i_k) du_k\Big]$$

$$= \mathbb{E}\Big[\int \nabla_\theta \mu_\theta(u_k|i_k) Q_\theta(i_k,u_k) du_k\Big] - \mathbb{E}\Big[\nabla_\theta \int \mu_\theta(u_k|i_k) du_k \, b(i_k)\Big]$$

$$= \mathbb{E}\Big[\int \nabla_\theta \mu_\theta(u_k|i_k)\Big[Q_\theta(i_k,u_k)\Big]du_k\Big]$$

Hence subtracting a baseline does not change the policy gradient, i.e. the mean MC estimator will remain unchanged. But we can exploit $b(i_k)$ to reduce its variance.

Actor-critic methods

Choose the cost-to-go function as the baseline $J_\phi(i)$ and apply the REINFORCE trick:

$$\nabla_\theta J_\theta(i) = \sum_{k=0}^{N-1} \mathbb{E}_{p(i_k)\mu_\theta(u_k|i_k)} \left[\nabla_\theta \log \mu_\theta(u_k|i_k) \Big(Q_\theta(i_k, u_k) - J_\psi(i_k) \Big) \right]$$

The corresponding objective for a single sampled sequence is then

$$\theta^*, \psi^* := \arg\min_{\theta,\psi} \sum_{k=0}^{N-1} \Big[C_k - J_\psi(i_k) \Big] \log \mu(u_k|i_k).$$

Doing bootstrapping and replacing $C_k - J_\psi(i_k)$ by one-step TD error $\delta_k = g(i_k, u_k, i_{k+1}) + J_\psi(i_{k+1}) - J_\psi(i_k)$ gives objective

$$\theta^*, \psi^* := \arg\min_{\theta,\psi} \sum_{k=0}^{N-1} \delta_k \log \mu(u_k|i_k).$$

A generic on-policy actor-critic algorithm

repeat ▷ episodes

  for do $k = 0 \to N-1$ ▷ go forward in time

    act $u_k \sim \mu(u_k|i_k)$ and observe i_{k+1} ▷ inside the loop now

    $\delta_k := g(i_k, u_k, i_{k+1}) + \alpha J_\psi(i_{k+1}) - J_\psi(i_k)$

    $\theta := \theta - \gamma_a \delta_k \nabla_\theta \log \mu_\theta(u_k|i_k)$

    $\psi := \psi - \gamma_c \delta_k \nabla_\psi J_\psi(i_k)$

  end for

until convergence

Extension of the above algorithm to multi-step bootstrapping is trivial.

Pros and cons of actor-critic methods

Pros:

i) Reduce the variance of the policy-gradient estimator
ii) Lift the need by whole-episode backups
iii) Applicable to non-episodic setups

Cons:

i) Bootstrapping causes bias
ii) Increased parameter count

Modern problems of AC approaches

- How to generalize to off-policy?
- How to permit benefiting from a replay memory?
- How to control policy update speed?
  - Baselines reduce estimator variance but do not determine training speed.
  - Policies are distributions, but are improved via parameter updates.
  - The magnitude of a parameter update does not give sufficient clue about the magnitude of the change on the policy distribution.
  - Online RL requires slow policy updates for stable training.
  - But how to quantify change for distributions? We need information theory for this.

Information

We would like to **measure** the amount of information received when a binary variable $x \in \{0, 1\}$ is observed.

Information: Degree of surprise after observing x.

Devise a function $h(x)$ to quantify information gained from x.

How should $h(x)$ look like?

When we observe two independent binary variables x and y, the information received should be the sum of the individual events.

Because independence implies $p(x, y) = p(x)p(y)$, it is suitable to measure information by

$$h(x) = -\log_2 p(x).$$

Base 2 is arbitrary, except having historical roots at communication theory. When base 2 is used, the measure is called a *bit*!

Negative sign assures that information with surprise, i.e. occurrence of a low-probability event.

Entropy

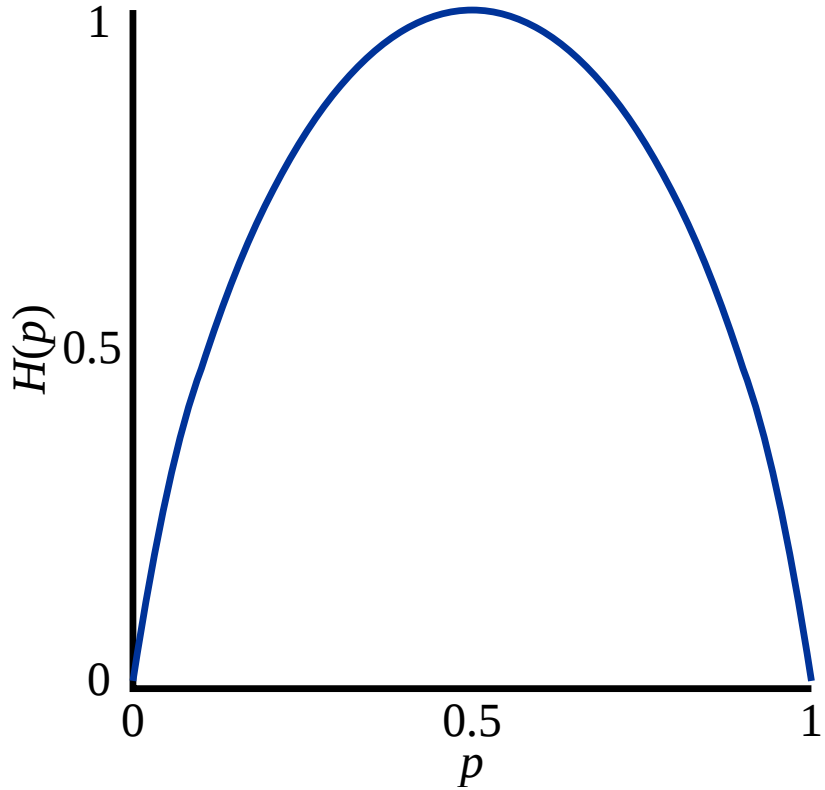
Expected amount of information for random variable x living in a sample space $\mathcal{X}$ and following a distribution $p(x)$:

$$H[x] = -\sum_{x \in \mathcal{X}} \log_2 p(x) p(x).$$

Note that the case for $p(x) = 0$ looks degenerate. Handle this by $\lim_{p \to 0} p \ln p = 0$, hence $H[x] = 0$.

For binary random variables:

$$H(p) = -p \log_2 p - (1-p) \log_2 (1-p).$$



Examples

Example 1: Consider the case where we have four possible states. When they are equally likely, the entropy turns out to be

$$H[x] = 4 \times \Big[-\frac{1}{4} \log_2 \frac{1}{4} \Big] = 2 \text{ bits.}$$

Example 2: Assume we have again four possible states, this time with probabilities $\Big(\frac{5}{8}, \frac{1}{4}, \frac{1}{16}, \frac{1}{16}\Big)$. Then the entropy is

$$\begin{aligned} H[x] &= -\frac{5}{8} \log_2 \frac{5}{8} - \frac{1}{4} \log_2 \frac{1}{4} - 2\frac{1}{16} \log_2 \frac{1}{16} \\ &= 0.42 + 0.5 + 0.5 = 1.42 \text{ bits.} \end{aligned}$$

There is more information in the uniform case!

Measures of information content

- Measures of information content

$$-\log_2 p(x) \rightarrow \text{bits}$$
$$-\ln p(x) \rightarrow \text{nats}$$

- Distributions that maximize the entropy
  - Discrete $\rightarrow$ uniform
  - Continuous (for a given location and spread) $\rightarrow$ normal!

Entropy in the continuous domain

- No exact counterpart.
- Using mean value theorem, we attain that entropy of a continuous density $p(x)$ differs from the term below by $-\ln \Delta$

$$H[x] = -\int p(x) \log p(x) dx.$$

  This term is called the **differential entropy**.
- Although differential entropy diverges from the exact entropy as $\Delta \to 0$, it is often used in place of the plain entropy for continuous densities. We will adopt the same convention here.

Relative entropy or KL divergence

Suppose for some reason, we need to approximate $p(x)$ by another density $q(x)$, which has some more pleasant properties. The **additional information** (in nats) required to be conveyed as a result of using $q(x)$ in place of $p(x)$ is

$$-\log q(x) - \Big(-\log p(x)\Big) = -\log \frac{q(x)}{p(x)} = \log \frac{p(x)}{q(x)}.$$

Since x follows $p(x)$, the expected additional information is

$$D_{KL}[p||q] = \int \log \frac{p(x)}{q(x)} p(x) dx.$$

This quantity is called *relative entropy* or **Kullback-Leibler divergence** and denoted by $D_{KL}[p||q]$.

KL divergence is a statistical distance

Considering that $-\log x$ is a convex function,

$$D_{KL}[p||q] = -\int p(x) \log \frac{q(x)}{p(x)} dx$$
$$\geq -\log \underbrace{\int p(x) \frac{q(x)}{p(x)} dx}_{1} = 0.$$

Because $-\log x$ is a *strictly* convex function (i.e. equality holds only at intersection points),

$$p(x) = q(x) \iff D_{KL}[p||q] = 0.$$

Hence, KL divergence is a statistical distance measure between two distributions. Note that $D_{KL}[p||q] \neq D_{KL}[q||p]$.

Other distance metrics

Total variation distance

$$D_{TV}(p||q) = \max_x |p(x) - q(x)|.$$

Also known as the maximum norm $||p - q||_\infty$ for discrete p and q.

Hellinger distance

$$D_H(p||q) = \sqrt{\frac{1}{2}\int \left(\sqrt{\tilde{p}(x)} - \sqrt{\tilde{q}(x)}\right)^2 \lambda(dx)}$$

for measure λ that satisfies $p(dx) = \tilde{p}(x)\lambda(dx)$ and $q(dx) = \tilde{q}(x)\lambda(dx)$. Here p and q are called the **Radon-Nikodym derivatives** of p and q. Some properties:

$$D_{TV}(p||q) \leq \sqrt{\frac{1}{2}D_{KL}(p||q)}$$

Pinsker's inequality

$$D_H(p||q)^2 \leq D_{TV}(p||q) \leq \sqrt{2}D_H(p||q)$$

Trust region methods

Vanilla AC optimization problem can be expressed as

$$\arg\min_{\theta} \sum_{k=0}^{N-1} \mathbb{E}_{p(i_k)\mu_\theta(u_k|i_k)} \left[\log \mu_\theta(u_k|i_k) A_\mu(i,u) \right]$$

where $A_\mu(i,u) := Q_\mu(i,u) - J_\mu(i)$ is the **advantage function**.
Define visitation frequency $\rho_\mu(i) = \sum_{k=0}^{\infty} \alpha^k \mathbb{P}(i_k = i|\mu)$. Then provably

$$\begin{aligned}
J_{\mu'} &= J_\mu + \mathbb{E}_{\mu'} \left[\sum_{k=0}^{\infty} \alpha^k A_\mu(i_k, u_k) \right] \\
&= J_\mu + \sum_{k=0}^{\infty} \sum_{i} \mathbb{P}(i_k = i|\mu') \sum_{u} \mu'(u|s) \alpha^k A_\mu(i,u) \\
&= J_\mu + \sum_{i} \sum_{k=0}^{\infty} \alpha^k \mathbb{P}(i_k = i|\mu') \sum_{u} \mu'(u|i) A_\mu(i,u) \\
&= J_\mu + \sum_{i} \rho_{\mu'}(i) \sum_{u} \mu'(u|i) A_\mu(i,u).
\end{aligned}$$

The outcome

$$J_{\mu'} = J_\mu + \sum_i \rho_{\mu'}(i) \sum_u \mu'(u|s) A_\mu(i,u).$$

means a policy update $\mu \to \mu'$ guarantees improvement if $\sum_u \mu'(u|s) A_\mu(i,u) \leq 0, \forall i$. Then do

$$\arg\min_{\mu'} J_\mu + \sum_i \rho_{\mu'}(i) \sum_u \mu'(u|s) A_\mu(i,u).$$

The dependency of the objective on μ via ρ makes optimization difficult. Use instead the objective below

$$L_\mu(\mu') := J_\mu + \sum_i \rho_\mu(i) \sum_u \mu'(u|s) A_\mu(i,u).$$

Policy update with improvement guarantee

If the policy is updated by $\mu' = \gamma\Big(\arg\min_{\mu^*} L_\mu(\mu^*)\Big) + (1-\gamma)\mu$ then

$$J_{\mu'} \leq L_\mu(\mu') + \frac{2\alpha \min_i |\mathbb{E}_{\mu'}[A_\mu(i,u)]|}{(1-\alpha)^2}\gamma^2, \qquad \text{[Kakade-Langford]}$$

which means a guaranteed improvement. Here $\gamma \in (0,1)$ tunes the similarity of μ and μ'. Assuming

$$\gamma = D_{TV}^{\max}(\mu||\mu') = \max_i(D_{TV}(\mu(\cdot|i)||\mu'(\cdot|i)))$$

this bound can be generalized to arbitrary policies and update rules as

$$\begin{aligned} J_{\mu'} &\leq L_\mu(\mu') + \frac{2\alpha \min_{i,u} |A_\mu(i,u)|}{(1-\alpha)^2} D_{TV}^{\max}(\mu||\mu')^2 \\ &\leq L_\mu(\mu') + \frac{4\alpha \min_{i,u} |A_\mu(i,u)|}{(1-\alpha)^2} D_{KL}^{\max}(\mu||\mu')^2 := M(\mu') \qquad \text{[Pinsker]} \end{aligned}$$

where $D_{KL}^{\max}(\mu||\mu') = \max_i(D_{KL}(\mu(\cdot|i)||\mu'(\cdot|i)))$. An update $\mu \to \mu'$ gives $M(\mu') \leq M(\mu)$. As $M(\mu) = J_\mu \Rightarrow J_{\mu'} - J_\mu \leq M(\mu') - M(\mu)$, every update improves.

From theory to practice

The second term of $M(\mu)$ involves a minimization problem and may induce too slow policy updates. Replace by a trust region constraint

$$\arg\min_{\theta'} L_{\mu_\theta}(\theta')$$
$$\text{subject to } D_{KL}^{\max}(\mu_\theta||\mu_{\theta'}) \leq \tau$$

for trust region radius $\tau > 0$. Approximate

$$D_{KL}^{\max}(\mu||\mu') \approx \mathbb{E}_{i\sim\mu_\theta}[D_{KL}(\mu(\cdot|i)||\mu'(\cdot|i))]$$

and do importance sampling wrt $b(u|i)$, which gives the final objective

$$\arg\min_{\theta'} \mathbb{E}_{i\sim\rho_{\mu_\theta}, u\sim b(u|i)}\left[\frac{\mu_{\theta'}(u|i)}{b(u|i)} Q_{\mu_\theta}(i,u)\right]$$
$$\text{subject to } \mathbb{E}_{i\sim\rho_{\mu_\theta}}[D_{KL}(\mu(\cdot|i)||\mu'(\cdot|i))] \leq \tau$$

Common to choose $b(u|i) := \mu_\theta(u|i)$.

Trust Region Policy Optimization (TRPO)

repeat

act $u \sim \mu_\theta(u|i)$, observe i', add $\mathcal{D} := \mathcal{D} \cup (i, u, g_i, i')$ ▷ explore

$\widetilde{\mathcal{D}} \sim \mathcal{D}$ ▷ improve

Solve

$$\theta' := \arg\min_{\theta^*} \frac{1}{|\widetilde{\mathcal{D}}|} \sum_{i \in \widetilde{\mathcal{D}}} \frac{\mu_{\theta^*}(u|i)}{\mu_\theta(u|i)} (g_i + \alpha J_\psi(i') - J_\psi(i))$$

$$\text{subject to } \frac{1}{|\widetilde{\mathcal{D}}|} \sum_{i \in \widetilde{\mathcal{D}}} D_{KL}(\mu_\theta(u|i) || \mu_{\theta^*}(u|i))$$

$\psi := \psi - \gamma_c \frac{1}{|\widetilde{\mathcal{D}}|} \sum_{i \in \widetilde{\mathcal{D}}} (g_i + \alpha J_\psi(i') - J_\psi(i)) \nabla_\psi J_\psi(i)$

$\theta \leftarrow \theta'$

until convergence

Making TRPO really work

The KL term of TRPO simply restricts the magnitude of update but at the expense of many computational difficulties. One can do the same with only clipping:

$$\arg\min_{\theta'} \mathbb{E}_{i\sim\rho_{\mu_\theta}, u\sim\mu_\theta(u|i)}\left[\left|\frac{\mu_{\theta'}(u|i)}{\mu_\theta(u|i)}\right|_{\epsilon}^{1-\epsilon} Q_{\mu_\theta}(i,u)\right]$$

where the clipping operator is defined for some small $\epsilon > 0$ as

$$\left|t\right|_{\epsilon}^{1-\epsilon} = \min(\max(t,\epsilon), 1-\epsilon).$$

Proximal Policy Optimization (PPO)

The working version of TRPO.

repeat

act $u \sim \mu_\theta(u|i)$, observe i', add $\mathcal{D} := \mathcal{D} \cup (i, u, g_i, i')$ ▷ explore

$\widetilde{\mathcal{D}} \sim \mathcal{D}$ ▷ improve

Solve

$$\theta' := \arg\min_{\theta'} \mathbb{E}_{i\sim\rho_{\mu_\theta}, u\sim\mu_\theta(u|i)} \left[\left| \frac{\mu_{\theta'}(u|i)}{\mu_\theta(u|i)} \right|_{\epsilon}^{1-\epsilon} (g_i + \alpha J_\psi(i') - J_\psi(i)) \right]$$

$$\psi := \psi - \gamma_c \frac{1}{|\widetilde{\mathcal{D}}|} \sum_{i\in\widetilde{\mathcal{D}}} (g_i + \alpha J_\psi(i') - J_\psi(i)) \nabla_\psi J_\psi(i)$$

$\theta \leftarrow \theta'$

until convergence

PPO Results

Taken on the MuJoCo Half-Cheetah environment. Clipping works better when used alone compared to together with the KL term.

algorithm	avg. normalized score
No clipping or penalty	-0.39
Clipping, $\epsilon = 0.1$	0.76
Clipping, $\epsilon = 0.2$	**0.82**
Clipping, $\epsilon = 0.3$	0.70
Adaptive KL $d_{\text{targ}} = 0.003$	0.68
Adaptive KL $d_{\text{targ}} = 0.01$	0.74
Adaptive KL $d_{\text{targ}} = 0.03$	0.71
Fixed KL, $\beta = 0.3$	0.62
Fixed KL, $\beta = 1.$	0.71
Fixed KL, $\beta = 3.$	0.72
Fixed KL, $\beta = 10.$	0.69

Figure: `https://arxiv.org/abs/1707.06347`

Off-policy policy-gradient methods

- The stable use of methods introduced thus far is on-policy.
- On-policy algorithms require new samples for parameter update.
- Off-policy algorithms can reuse past experience, hence are often more sample efficient.
- Complementing Q-learning with a parameterized actor net is a naive solution, that results in unstable coordinate-descent updates.
- Introducing **target networks** that mimic the originals but are updated at slower pace delivers a working solution.

Deep Deterministic Policy Gradient (DDPG)

Remember the Bellman equation

$$Q_\psi(i,u) = \mathbb{E}_{i'}\Big[g_i + \alpha \mathbb{E}_{\mu_\theta(u'|i')}[Q_\psi(i',u')]\Big].$$

If $\mu(u'|i')$ is **deterministic**, the equation boils down to

$$Q_\psi(i,u) = \mathbb{E}_{i'}\Big[g_i + \alpha Q_\psi(i', \mu_\theta(u'|i'))\Big].$$

Hence solve

$$\theta^*, \psi^* := \arg\min_{\theta,\psi} \mathbb{E}\Big[\Big(g_i + \alpha Q_\psi(i', \mu_\theta(u'|i')) - Q_\psi(i, \mu(u|i))\Big)^2\Big].$$

Innovation: Making a good old idea work

Do the following for computational stability

- Treat $Q_\psi(i', \mu_\theta(u'|i'))$ as constant wrt ψ, θ.
- Maintain a replay buffer (first time for AC methods!)
- Use $b(u|i) := \mu(u|i) + \omega$ as the behavior policy where ω is an **Ornstein-Uhlenbeck (OU)** process,

$$W_k \sim \mathcal{N}(0, dt),$$
$$d\omega_{k+1} = -0.15\omega_k dt + 0.2 dW_k,$$

  and dt is the time difference between steps k and $k+1$.
- Chain rule applies as policy is deterministic. The policy gradient is

$$\nabla_\theta J = \mathbb{E}_{\rho_b(i)}[\nabla_u Q_\psi(i, u) \nabla_\theta \mu(u|i)].$$

- Use target networks θ', ψ' to stabilize coordinate descent. Update them with time lag: $\theta' := \tau\theta + (1-\tau)\theta'$ and $\psi' := \tau\psi + (1-\tau)\psi'$ for $\tau \in (0, 1)$. This technique is called **Polyak averaging**.

The DDPG algorithm

repeat

$W \sim \mathcal{N}(0, dt), \omega := -0.15\omega dt + 0.2W$ ▷ sample OU

act $u \sim \mu_\theta(u|i) + \omega$, observe i', add $\mathcal{D} := \mathcal{D} \cup (i, u, g_i, i')$ ▷ explore

$\widetilde{\mathcal{D}} \sim \mathcal{D}$ ▷ sample minibatch

$y_i := g_i + \alpha Q_{\psi'}(i', \mu_{\theta'}(\cdot|i')), \qquad \forall (i, g_i, i') \in \widetilde{\mathcal{D}}$ ▷ compute targets

$\psi := \arg\min_\psi \frac{1}{|\widetilde{\mathcal{D}}|} \sum_{(i,u)\in\widetilde{\mathcal{D}}} \Big(y_i - Q_\psi(i, u)\Big)^2$ ▷ update critic

$\theta := \arg\min_\theta \frac{1}{|\widetilde{\mathcal{D}}|} \sum_{i\in\widetilde{\mathcal{D}}} Q_\psi(i, \mu_\theta(\cdot|i))$ ▷ update actor

$\theta' := \tau\theta + (1-\tau)\theta', \qquad \psi' := \tau\psi + (1-\tau)\psi'$ ▷ update target nets

until convergence

Huge improvement over methods of the same sort in terms of numerical stability. However, DDPG is still not sufficiently sample inefficient and computationally stable.

Soft Actor Critic (SAC)

- Goal is to stabilize training by permitting a **stochastic** policy network.
- First to apply the maximum-entropy technique to off-policy actor-critic RL. It has been used in other contexts before such as Inverse RL and model-based RL (Guided Policy Search).
- This method is currently **the state of the art** of model-free policy-gradient methods.
- **Maximum Entropy RL:**

$$J_{\psi}^{\mu} = \sum_{k=0}^{\infty} \mathbb{E}_{\rho_{\mu}(i_k)}[g_k + H(\mu(\cdot|i_k))]$$

Dual goals: i) minimize cost-to-go, ii) maximize policy entropy.

Theoretical guarantees of SAC

Soft Bellman backup operator:

$$(TJ_\mu)(i) = g_i + \alpha \mathbb{E}_{i'}[\mathbb{E}_{u' \sim \mu}[Q_\mu(i', u') - \log \mu(u'|i')]]$$

Soft policy evaluation lemma: $\lim_{k \to \infty} T_\mu^k Q = Q_\mu$. Proof follows from redefining cost as $g_i + H(\mu(\cdot|i_k))$.

Soft policy update rule:

$$\mu' := \arg\min_{\mu' \in \Pi} D_{KL}\Big(\mu'(\cdot|i) \Big|\Big| \exp(-Q_\mu(i,u)) \Big/ \sum_{u \in U} \exp(-Q_\mu(i,u))\Big)$$

Soft policy improvement theorem: Repeating soft policy evaluation and soft policy update rule converges to the optimal policy within set Π after finite iterations.

The SAC algorithm

repeat

act $u \sim \mu_\theta(u|i)$, observe i', add $\mathcal{D} := \mathcal{D} \cup (i, u, g_i, i')$

$\widetilde{\mathcal{D}} \sim \mathcal{D}$

$u' \sim \mu_\theta(u'|i'), \qquad \forall (g_i, i') \in \widetilde{\mathcal{D}}$ ▷ compute targets

$y_i := g_i + \alpha \max_{j=0,1} Q_{\psi'_j}(i', u') - \log \mu_\theta(u'|i')$

$\psi_j := \arg\min_{\psi_j} \frac{1}{|\widetilde{\mathcal{D}}|} \sum_{(i,u)\in\widetilde{\mathcal{D}}} \left(y_i - Q_{\psi_j}(i,u)\right)^2, \qquad j = 0, 1$

$\tilde{\epsilon}_i \sim p(\epsilon), \forall i \in \widetilde{\mathcal{D}}$

$\theta := \arg\min_\theta \frac{1}{|\widetilde{\mathcal{D}}|} \sum_{i\in\widetilde{\mathcal{D}}} \max_{j=0,1} Q_{\psi_j}(i, f_\theta(i, \tilde{\epsilon}_i)) - \log \mu_\theta(f_\theta(i, \tilde{\epsilon}_i)|i)$

$\psi'_j := \tau\psi_j + (1-\tau)\psi'_j, \qquad j = 0, 1$

until convergence

Above $\tilde{\epsilon} \sim p(\epsilon), \tilde{u} = f_\theta(i, \tilde{\epsilon})$ is a **reparameterization** of $\mu_\theta(\cdot|i)$, i.e. it follows the same distribution as $\tilde{u} \sim \mu_\theta(\cdot|i)$.

SAC results

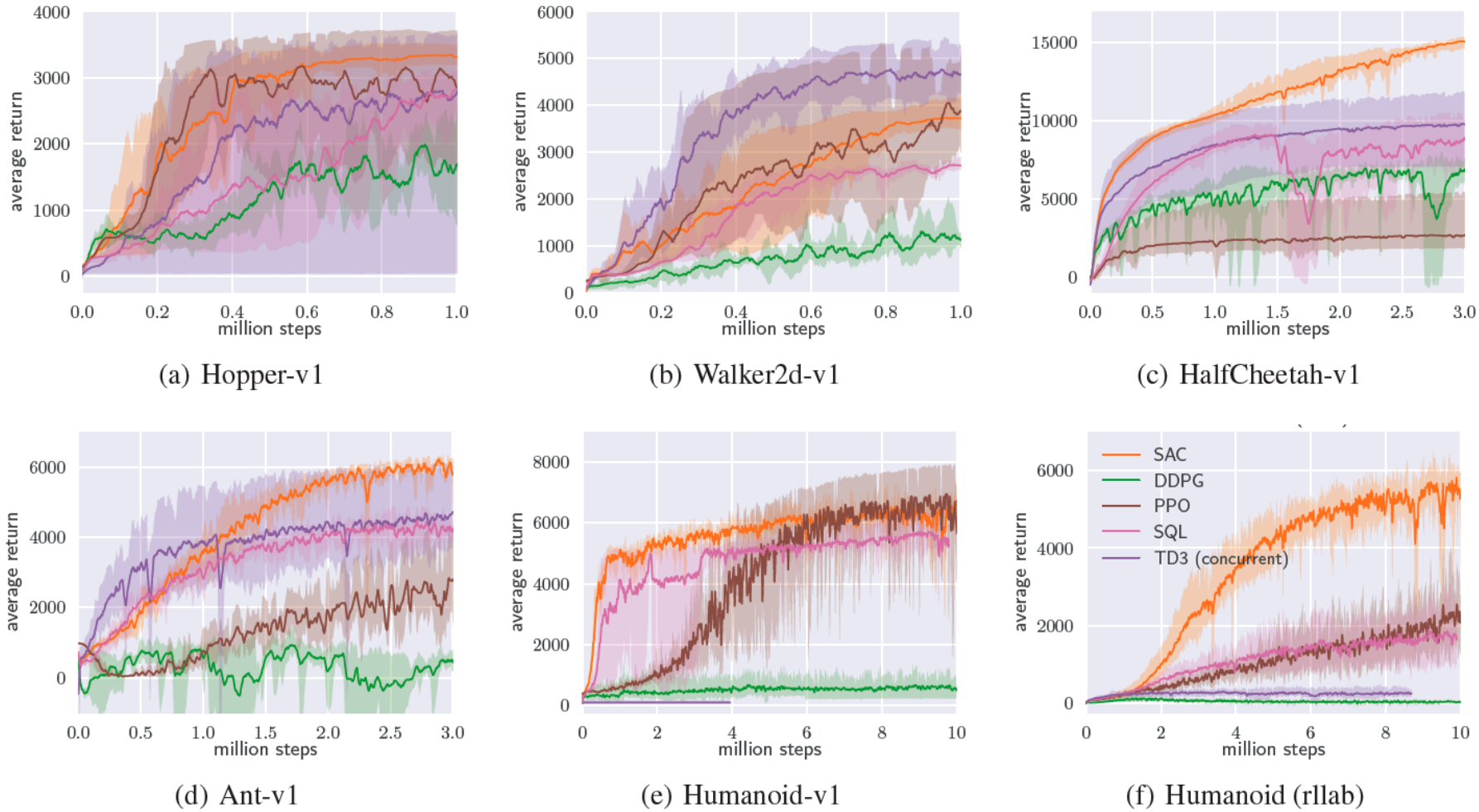


Figure: https://arxiv.org/abs/1801.01290