Deep Reinforcement Learning based on Policy Gradients

Ângelo G. Lovatto
angelo.lovatto@usp.br

June 27, 2019

INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
UNIVERSIDADE DE SÃO PAULO

Table of contents

Policy Gradients Introduction

Setup

Environment as a Markov Decision Process (MDP) [7]: $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rho_0)$.

- $\mathcal{S}$ is the set of possible states;
- $\mathcal{A}$ is the set of possible actions;
- $\mathcal{P}(s' \mid s, a)$ is the next state distribution;
- $\mathcal{R}(s, a, s')$ is the reward function;
- $\rho_0(s)$ is the initial state distribution.

Setup

A *trajectory* is a state-action sequence $\tau = (s_0, a_0, s_1 \ldots, a_{T-1}, s_T)$, whose probability under a stochastic policy π is given by

$$P(\tau \,|\, \pi) \doteq \rho_0(s_0) \prod_{t=0}^{T-1} \pi(a_t \,|\, s_t) \mathcal{P}(s_{t+1} \,|\, s_t, a_t),$$

and its undiscounted *return* is given by

$$R(\tau) \doteq \sum_{t=0}^{T-1} \mathcal{R}(s_t, a_t, s_{t+1})$$

The on-policy value functions are defined as usual:

$$V^\pi(s) \doteq \mathop{\mathbb{E}}_{\tau \sim \pi} [R(\tau) \,|\, s_0 = s], \quad Q^\pi(s, a) \doteq \mathop{\mathbb{E}}_{\tau \sim \pi} [R(\tau) \,|\, s_0 = s, a_0 = a].$$

Policy Gradient Methods - Problem

We consider a smoothly parameterized policy class:

- fix a set of parametric stochastic policies $\{\pi_\theta; \theta \in \mathbb{R}^n\}$
- we assume $\nabla_\theta \pi_\theta(a \mid s), (s, a) \in \mathcal{S} \times \mathcal{A}$, exists and is finite

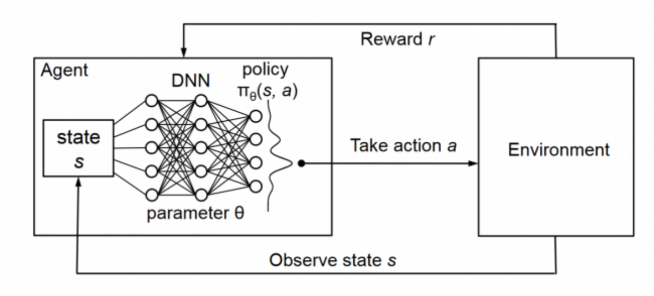


Figure source: Mao et. al[4]

Policy Gradient Methods - Problem

We'd like to find the parameters which maximize the expected return

$$\theta^* = \arg\max_\theta J(\theta), \text{ where } J(\theta) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[R(\tau)].$$

Policy gradient methods optimize the policy using gradient ascent, e.g.,

$$\theta_{k+1} = \theta_k + \alpha \nabla J(\theta_k).$$

Problems:

- how to derive the gradient from the expectation?
- how to obtain sample estimates of the gradient?

Score Function Estimator

Suppose that x is a random variable, f is a function, and we are interested in computing

$$\frac{\partial}{\partial\theta}\mathbb{E}_x\left[f(x)\right], \quad x \sim p(\cdot;\theta).$$

The *score function* (SF) estimator of the equation above is derived as follows:

$$\begin{aligned}\frac{\partial}{\partial\theta}\mathbb{E}_x\left[f(x)\right] &= \frac{\partial}{\partial\theta}\int p(x;\theta)f(x)dx = \int \frac{\partial}{\partial\theta}p(x;\theta)f(x)dx \\ &= \int p(x;\theta)\frac{\partial}{\partial\theta}\log p(x;\theta)f(x)dx = \mathbb{E}_x\left[f(x)\frac{\partial}{\partial\theta}\log p(x;\theta)\right].\end{aligned}$$

Policy Gradient Derivation

The score function estimator for $J(\theta)$ is then

$$
\begin{aligned}
\nabla J(\theta) &= \nabla_\theta \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} [R(\tau)] \\
&= \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} [R(\tau) \nabla_\theta \log P(\tau \mid \theta)] \,,
\end{aligned}
$$

where

$$
\begin{aligned}
\nabla_\theta \log P(\tau \mid \theta) &= \nabla_\theta \left(\rho_0(s_0) + \sum_{t=0}^{T-1} \left(\log \pi_\theta(a_t \mid s_t) + \log \mathcal{P}(s_{t+1} \mid s_t, a_t) \right) \right) \\
&= \sum_{t=0}^{T-1} \underbrace{\nabla_\theta \log \pi_\theta(a_t \mid s_t)}_{\text{no dynamics model needed!}} \quad .
\end{aligned}
$$

Thus,

$$
\nabla J(\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T-1} R(\tau) \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right]
$$

A Simple Policy Gradient

We can then estimate the gradient by averaging across sampled trajectories:

$$\nabla J(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{T-1} R(\tau_i) \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t}).$$

Intuitively, taking a step in the gradient direction

- increases the probability of trajectories with positive $R(\tau)$,
- decreases the probability of trajectories with negative $R(\tau)$.

Reducing Variance: Temporal Structure

We can remove terms that don't depend on current action:

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} R(\tau)\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right] \\
&= \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \sum_{t'=0}^{T-1} \mathcal{R}(s_{t'+1} \mid s_{t'}, a_{t'})\right] \\
&= \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \left(\sum_{i=0}^{t-1} \mathcal{R}(s_{i+1} \mid s_i, a_i) + \sum_{j=t}^{T-1} \mathcal{R}(s_{j+1} \mid s_j, a_j)\right)\right] \\
&= \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \underbrace{\sum_{t'=t}^{T-1} \mathcal{R}(s_{t'+1} \mid s_{t'}, a_{t'})}_{\text{reward-to-go}}\right]
\end{aligned}
$$

Reducing Variance: Adding a Baseline

Furthermore, a state-dependent function can be subtracted

$$g \approx \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t}) \left(\sum_{t'=t}^{T-1} \mathcal{R}(s_{i,t'+1} \mid s_{i,t'}, a_{i,t'}) - b(s_{i,t}) \right).$$

which is also known as a *control variate*. It can be shown that

$$\mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) b(s_t) \right] = 0,$$

thus, the estimate is still unbiased. This form of is also known as the REINFORCE estimator.

Vanilla Policy Gradient and GAE

General Policy Gradient

In general, the policy gradient can be expressed as

$$g = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T-1} \Phi_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right]$$

where Φ_t may be one of the following.

- $R(\tau)$
- $\sum_{k=t}^{T-1} \mathcal{R}(s_{k+1} \mid s_k, a_k)$
- $\sum_{k=t}^{T-1} \mathcal{R}(s_{k+1} \mid s_k, a_k) - b(s_t)$
- $Q^{\pi_\theta}(s_t, a_t)$
- $A^{\pi_\theta}(s_t, a_t)$
- $r_{t+1} + V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t)$

The advantage function, $A^{\pi_\theta}(s_t, a_t) = Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)$, yields very low variance in practice.

Schulman et al., Generalized Advantage Estimation [6]

Policy Gradient w/ Advantage Function

$$g = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T-1} A^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right]$$

Intuitively, taking a step in the direction of this gradient:

- increases the probability of better-than-average actions;
- decreases the probability of worse-than-average actions.

We don't have access to the true value functions and sampled rewards have high variance. We can address this in two ways:

- using discounted returns
- using a learned value function (critic)

Reducing Variance: Discounting

Consider discounted variants of the value functions

$$V^{\pi,\gamma}(s) \doteq \mathop{\mathbb{E}}_{\tau\sim\pi}\left[R^{\gamma}(\tau) \,|\, s_0 = s\right],$$
$$Q^{\pi,\gamma}(s,a) \doteq \mathop{\mathbb{E}}_{\tau\sim\pi}\left[R^{\gamma}(\tau) \,|\, s_0 = s, a_0 = a\right],$$
$$A^{\pi,\gamma}(s,a) = Q^{\pi,\gamma}(s,a) - V^{\pi,\gamma}(s),$$

where $R^{\gamma}(\tau) \doteq \sum_{t=0}^{T-1} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1})$.

Reducing Variance: Discounting

Consider discounted variants of the value functions

$$V^{\pi,\gamma}(s) \doteq \mathop{\mathbb{E}}_{\tau\sim\pi}\left[R^{\gamma}(\tau) \,|\, s_0 = s\right],$$

$$Q^{\pi,\gamma}(s,a) \doteq \mathop{\mathbb{E}}_{\tau\sim\pi}\left[R^{\gamma}(\tau) \,|\, s_0 = s, a_0 = a\right],$$

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

where $R^{\gamma}(\tau) \doteq \sum_{t=0}^{T-1} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1})$.

Note that $A^{\pi,\gamma}(s,a)$ is equivalent to the *TD residual,*

$$A^{\pi,\gamma}(s,a) = \mathbb{E}\left[r_{t+1} + \gamma V^{\pi,\gamma}(s_{t+1}) - V^{\pi,\gamma}(s_t) \,|\, s_t = s, a_t = a\right].$$

Reducing Variance: Bootstrapping

In fact, any n-step TD residual is valid.

$$
\begin{aligned}
&A^{\pi,\gamma}(s,a) \\
&\quad = \mathbb{E}\left[r_{t+1} + \gamma V^{\pi,\gamma}(s_{t+1}) - V^{\pi,\gamma}(s_t) \,|\, s_t = s, a_t = a\right] \\
&\quad = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 V^{\pi,\gamma}(s_{t+2}) - V^{\pi,\gamma}(s_t) \,|\, s_t = s, a_t = a\right] \\
&\quad = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 V^{\pi,\gamma}(s_{t+3}) - V^{\pi,\gamma}(s_t) \,|\, s_t = s, a_t = a\right] \\
&\quad \vdots \\
&\quad = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-1} r_T - V^{\pi,\gamma}(s_t) \,|\, s_t = s, a_t = a\right]
\end{aligned}
$$

Reducing Variance: Bootstrapping

In practice, we use sample estimates and learned critics

$$
\begin{aligned}
\hat{A}_t^{(1)} &= r_{t+1} + \gamma V_\phi(s_{t+1}) - V_\phi(s_t) \\
\hat{A}_t^{(2)} &= r_{t+1} + \gamma r_{t+2} + \gamma^2 V_\phi(s_{t+2}) - V_\phi(s_t) \\
\hat{A}_t^{(3)} &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 V_\phi(s_{t+3}) - V_\phi(s_t) \\
&\vdots \\
\hat{A}_t^{(T)} &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-1} r_T - V_\phi(s_t)
\end{aligned}
$$

Analogous to the n-step return $G_t^{(n)}$ used in *TD* methods.

Generalized Advantage Estimation

The generalized advantage estimator is defined as follows

$$\begin{aligned}
\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} \doteq & \,(r_{t+1} + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)) && (1-\lambda) \\
& + (r_{t+1} + \gamma r_{t+2} + \gamma^2 V_\phi(s_{t+2}) - V_\phi(s_t)) && (1-\lambda)\lambda \\
& + (r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 V_\phi(s_{t+3}) - V_\phi(s_t)) && (1-\lambda)\lambda^2 \\
& \vdots \\
& + (r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-1} r_T - V_\phi(s_t)) && \lambda^{T-t-1}
\end{aligned}$$

Analogous to the lambda return G_t^λ used in $TD(\lambda)$ (forward-view).

GAE: Limiting Cases

It turns out that $\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)}$ can be computed as follows

$$\hat{A}_t^{GAE(\gamma,\lambda)} = \sum_{l=0}^{T-1} (\gamma\lambda)^l \delta_{t+l}^V, \text{ where } \delta_t^V = r_{t+1} + \gamma V(s_{t+1}) - V(s_t).$$

We can recover the one-step TD estimator and the monte carlo by setting λ to 0 and 1 respectively:

$$\begin{aligned} \hat{A}_t^{GAE(\gamma,0)} &= \delta_t^V &&= r_{t+1} + \gamma V(s_{t+1}) - V(s_t), \\ \hat{A}_t^{GAE(\gamma,1)} &= \sum_{l=0}^{T-1} \gamma^l \delta_{t+l}^V &&= R_t^\gamma - V(s_t). \end{aligned}$$

As in $TD(\lambda)$, λ makes a compromise between bias and variance.

Vanilla Policy Gradient w/ GAE

Algorithm 1 Vanilla Policy Gradient + GAE

```
Initialize policy parameter $\theta_0$ and value function parameter $\phi_0$
for k=0, 1, 2, ... do
    Collect $N$ transitions $(s_t, a_t, r_{t+1})$ following $\pi_{\theta_k}$
    Compute $\delta_t^V$ at all timesteps $t \in \{1, 2, \ldots, N\}$
    Compute $\hat{A}_t = \sum_{l=0}^{T-1} (\gamma\lambda)^l \delta_{t+l}^V$ for every timestep
    Fit baseline by minimizing $\|V_\phi(s_t) - R_t^\gamma\|^2$
    Compute $\theta_{k+1}$ with VPG update
end for
```

REINFORCE vs VPG w/ GAE

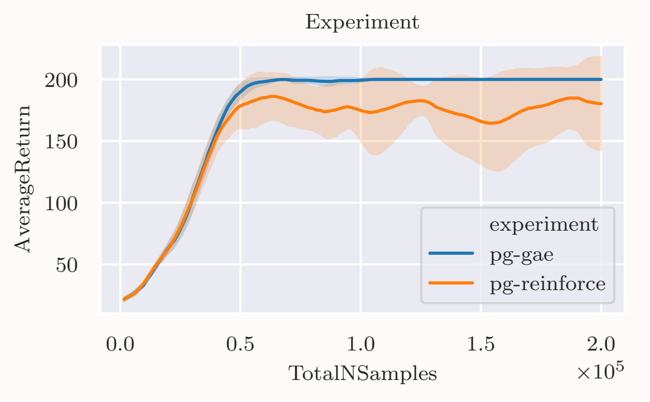


Figure 1: Learning curves in the CartPole-v0 environment, where REINFORCE uses a baseline of zero and VPG uses GAE(0.99,0.97)

Natural Policy Gradient

Problems with the Vanilla Gradient

$$g = \underset{\tau \sim \pi_\theta}{\mathbb{E}} \left[\sum_{t=0}^{T-1} A^{\pi_\theta}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right]$$

- The data distribution depends on the parameters θ
- Most optimizers work well in settings with i.i.d. data
- Hard to choose step size α
    - too big $\rightarrow$ bad policy $\rightarrow$ data collected under bad policy $\rightarrow$ collapse in performance
    - too small: inefficient use of data collected

Problems with the Vanilla Gradient

Consider a family of policies with parametrization:

$$\pi_\theta(a) = \begin{cases} \sigma(\theta) & a = 1 \\ 1 - \sigma(\theta) & a = 2 \end{cases}$$

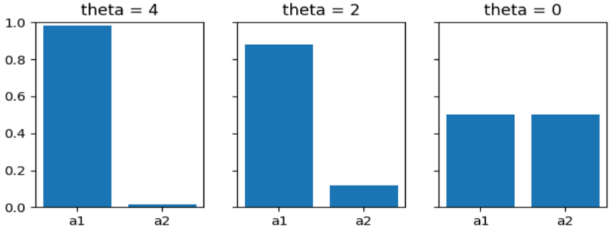


Figure: Small changes in the policy parameters can unexpectedly lead to **big** changes in the policy.

Figure source: `http://www.andrew.cmu.edu/course/10-703/`

Gradient Ascent in Parameter Space

The step size in gradient ascent comes from solving the following optimization problem, e.g., using line search

$$d^* = \arg\max_d J(\theta + d)$$
$$\text{s.t.} \quad \|d\| \leq \epsilon$$

However, the distance in parameters, as we've seen, gives no insight into the resulting policy change.

Gradient Ascent in Distribution Space

The step size in natural gradient ascent comes from solving the following optimization problem,

$$d^* = \arg\max_{d} J(\theta + d)$$
$$\text{s.t.} \quad D(\pi_\theta \| \pi_{\theta+d}) \leq \epsilon$$

It is be better to define the constraint in terms of a ”distance” between successive policies.

Measuring Dissimilarities between Distributions

The Kullback-Leibler (KL) divergence provides a way to measure dissimilarities between distributions.

$$D_{\mathrm{KL}}\left(P \,\|\, Q\right) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

$$D_{\mathrm{KL}}\left(p \,\|\, q\right) = \int_{-\infty}^{+\infty} p(x) \log \frac{p(x)}{q(x)} dx$$

We define the KL divergence between policies as follows.

$$\overline{D}_{\mathrm{KL}}\left(\theta_k \,\|\, \theta\right) \doteq \mathop{\mathbb{E}}_{s \sim \pi_{\theta_k}} \left[D_{\mathrm{KL}}\left(\pi_{\theta_k}(\cdot \mid s) \,\|\, \pi_\theta(\cdot \mid s)\right)\right].$$

Natural Policy Gradient: Problem

The objective then becomes

$$\theta_{k+1} = \arg\max_{\theta} J(\theta_k)$$
$$\text{s.t. } \overline{D}_{\text{KL}}\left(\theta_k \,\|\, \theta\right) \leq \delta,$$

- Easier to pick the distance threshold
- Invariant to the parameterization of the policy
- Hard to calculate in practice

Natural Policy Gradient: Solution

Using first- and second-order Taylor expansions for the objective and constraint respectively yields

$$\theta_{k+1} = \arg\max_{\theta} J(\theta_k) + g^{\intercal}(\theta - \theta_k)$$
$$\text{s.t.} \quad \tfrac{1}{2}(\theta - \theta_k)^{\intercal} F(\theta_k)(\theta - \theta_k) \leq \delta.$$

where $F(\theta_k)$ denotes the Hessian of the KL divergence, also known as the Fisher Information Matrix,

$$F(\theta_k) = \nabla_{\theta}^2 \overline{D}_{\mathrm{KL}}\left(\theta_k \,\|\, \theta\right)\big|_{\theta=\theta_k}.$$

A Lagrange multiplier argument gives the solution to this constrained optimization problem:

$$\theta_{k+1} = \theta_k + \underbrace{\sqrt{\frac{2\delta}{g^{\intercal} F(\theta_k)^{-1} g}}}_{\text{scaling term}} \underbrace{F(\theta_k)^{-1} g}_{\text{npg}},$$

Computing the Natural Gradient

Computing $F(\theta_k)^{-1}g$ can be expensive:

- DNN policy can have thousands of parameters
- Storing and inverting the Hessian becomes impractical

Computing the Natural Gradient

Computing $F(\theta_k)^{-1}g$ can be expensive:

- DNN policy can have thousands of parameters
- Storing and inverting the Hessian becomes impractical
- Can instead solve for $F(\theta_k)x = g$ using conjugate gradient
  - Does not form the whole matrix
  - Requires only a function $f(v) = F(\theta_k)v$

Computing the Natural Gradient

Computing $F(\theta_k)^{-1}g$ can be expensive:

- DNN policy can have thousands of parameters
- Storing and inverting the Hessian becomes impractical
- Can instead solve for $F(\theta_k)x = g$ using conjugate gradient
  - Does not form the whole matrix
  - Requires only a function $f(v) = F(\theta_k)v$
- We can evaluate $F(\theta_k)v$ using automatic differentiation.

$$F(\theta_k)v = \nabla_\theta \left(\left(\nabla_\theta \overline{D}_{\mathrm{KL}} \left(\theta_k \,\|\, \theta \right) \right)^{\mathsf{T}} v \right) \Big|_{\theta=\theta_k}.$$

NPG Algorithm

Algorithm 2 Natural Policy Gradient

Initialize policy parameter θ_0
for k=0, 1, 2, . . . **do**
    Collect a set of trajectories following π_{θ_k}
    Estimate $\hat{A}_t$'s using any advantage estimation algorithm
    Compute the policy gradient g
    Use CG and Fisher vector products to obtain $F(\theta_k)^{-1}g$
    Update the policy parameters

$$\theta_{k+1} = \theta_k + \sqrt{\frac{2\delta}{g^\intercal F(\theta_k)^{-1}g}} F(\theta_k)^{-1}g,$$

end for

Trust Region Policy Optimization

Resulting Policy Performance

Natural policy gradient gives us a good step size, however

- How do we ensure improvement in the objective?
- Can we estimate the performance of a new policy?

Relating Objectives of Two Policies

$$
\begin{aligned}
J(\tilde{\pi}) - J(\pi) &= J(\tilde{\pi}) - \mathop{\mathbb{E}}_{s_0 \sim \rho_0} \left[V^\pi(s_0) \right] \\
&= J(\tilde{\pi}) + \mathop{\mathbb{E}}_{\tau \sim \tilde{\pi}} \left[\sum_{i=1}^{T-1} V^\pi(s_i) - \sum_{j=0}^{T-1} V^\pi(s_j) \right] \\
&= J(\tilde{\pi}) + \mathop{\mathbb{E}}_{\tau \sim \tilde{\pi}} \left[\sum_{t=0}^{T-1} V^\pi(s_{t+1}) - V^\pi(s_t) \right] \\
&= \mathop{\mathbb{E}}_{\tau \sim \tilde{\pi}} \left[\sum_{t=0}^{T-1} \mathcal{R}(s_t, a_t) + V^\pi(s_{t+1}) - V^\pi(s_t) \right] \\
&= \mathop{\mathbb{E}}_{\tau \sim \tilde{\pi}} \left[\sum_{t=0}^{T-1} A^{\pi}(s_t, a_t) \right]
\end{aligned}
$$

Surrogate Objective Function

TRPO makes the following approximation to the result of $J(\tilde{\pi}) - J(\pi)$:

$$\mathop{\mathbb{E}}_{\tau \sim \tilde{\pi}} \left[\sum_{t=0}^{T-1} A^{\pi}(s_t, a_t) \right] \approx \mathop{\mathbb{E}}_{\tau \sim \pi} \left[\sum_{t=0}^{T-1} \frac{\tilde{\pi}(a_t \mid s_t)}{\pi(a_t \mid s_t)} A^{\pi}(s_t, a_t) \right].$$

That is, we approximate performance of a new policy relative to the old one using data gathered by the latter. In the case of parametric policies, the surrogate objective we seek to optimize is

$$\mathcal{L}(\theta_k, \theta) \doteq \mathop{\mathbb{E}}_{\tau \sim \pi_{\theta_k}} \left[\sum_{t=0}^{T-1} \frac{\pi_{\theta}(a_t \mid s_t)}{\pi_{\theta_k}(a_t \mid s_t)} A^{\pi_{\theta_k}}(s_t, a_t) \right].$$

TRPO: Problem

We could optimize $\mathcal{L}(\theta_k, \theta)$ freely, however

- we're substituting the trajectory distribution of the old policy for the new policy
- the approximation might not be accurate if the policies differ too much

It turns out that we can bound this approximation error

$$|J(\theta) - (\underbrace{J(\theta_k) + \mathcal{L}(\theta_k, \theta)}_{\approx J(\theta)})| \leq C\sqrt{\overline{D}_{\mathrm{KL}}\left(\theta_k \,\|\, \theta\right)}$$

TRPO: Problem

TRPO iteratively optimizes the surrogate objective function around a local neighborhood of the current policy, as measured by KL divergence:

$$\theta_{k+1} = \arg\max_{\theta} \mathop{\mathbb{E}}_{\tau \sim \pi_{\theta_k}} \left[\sum_{t=0}^{T-1} \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_k}(a_t \mid s_t)} A^{\pi_{\theta_k}}(s_t, a_t) \right]$$
$$\text{s.t. } \overline{D}_{\text{KL}}\left(\theta_k \,\|\, \theta\right) \leq \delta \, .$$

The constraint δ defines a *trust region* around our local approximation to the new policy's relative performance.

TRPO: Solution

It can be shown that $\nabla_\theta \mathcal{L}(\theta_k, \theta)$ evaluated at $\theta = \theta_k$ is equivalent to the policy gradient. Thus, the appropriate Taylor expansions give

$$\theta_{k+1} = \arg\max_\theta \, g^\intercal(\theta - \theta_k) \quad \text{s.t.} \quad \tfrac{1}{2}(\theta - \theta_k)^\intercal F(\theta_k)(\theta - \theta_k) \leq \delta,$$

yielding the same solution as in NPG. TRPO, however, tries to ensure

- improvement in the objective $J(\theta)$,
- satisfaction of the KL constraint.

Final update:

$$\theta_{k+1} = \theta_k + \alpha^j \sqrt{\frac{2\delta}{g^\intercal F(\theta_k)^{-1} g}} F(\theta_k)^{-1} g,$$

where $\alpha \in (0, 1)$ is a backtracking coefficient.

TRPO: Line Search

Algorithm 3 Line Search for TRPO

Compute proposed policy step $\Delta\theta = \sqrt{\frac{2\delta}{g^\intercal F(\theta_k)^{-1} g}} F(\theta_k)^{-1} g$
for $i = 0, 1, 2, \ldots, L$ **do**
  Compute proposed update $\theta = \theta_k + \alpha^i \Delta\theta$
  if $\mathcal{L}(\theta_k, \theta) \geq 0$ and $\overline{D}_{\mathrm{KL}}(\theta_k \,\|\, \theta) \leq \delta$ **then**
    accept the update and set $\theta_{k+1} = \theta_k + \alpha^i \Delta\theta$
    break
  end if
end for

TRPO: Algorithm

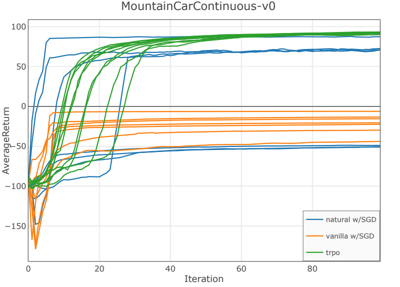
Algorithm 4 TRPO

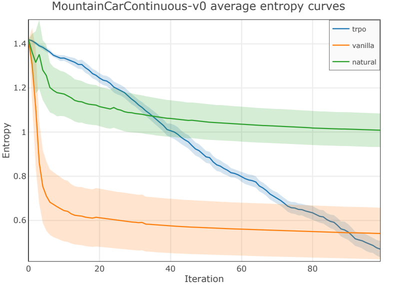
Initialize policy parameter θ_0
for k=0, 1, 2, . . . **do**
    Collect a set of trajectories following π_{θ_k}
    Estimate $\hat{A}_t$'s using any advantage estimation algorithm
    Compute the policy gradient g
    Use CG and Fisher vector products to obtain $F(\theta_k)^{-1}g$
    Compute the proposed update $\Delta\theta = \sqrt{\frac{2\delta}{g^\intercal F(\theta_k)^{-1}g}}F(\theta_k)^{-1}g$,
    Perform backtracking line search to obtain final update

$$\theta_{k+1} = \theta_k + \alpha^i \Delta\theta$$

end for

Comparisons





Questions?

Reward-to-go PG - Proof

Recall that

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} R(\tau)\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right] \\
&= \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \sum_{t'=0}^{T-1} \mathcal{R}(s_{t'+1} \mid s_{t'}, a_{t'})\right] \\
&= \sum_{t=0}^{T-1}\sum_{t'=0}^{T-1} \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\mathcal{R}(s_{t'+1} \mid s_{t'}, a_{t'})\right].
\end{aligned}$$

We'll show that, for $t' < t$ (when the reward comes before the action),

$$\mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\mathcal{R}(s_{t'+1} \mid s_{t'}, a_{t'})\right] = 0.$$

Reward-to-go PG - Proof

Let $f(t,t') = \nabla_\theta \log \pi_\theta(a_t \mid s_t)\mathcal{R}(s_{t'+1} \mid s_{t'}, a_{t'})$. Note that this term only depends on timesteps t, t' and $t'+1$. Thus, we can use the *marginal likelihood*

$$
\begin{aligned}
\mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[f(t,t')] &= \int_\tau P(\tau \mid \theta) f(t,t') \\
&= \int_{s_t,a_t,s_{t'},a_{t'},s_{t'+1}} P(s_t,a_t,s_{t'},a_{t'},s_{t'+1} \mid \theta) f(t,t') \\
&= \mathop{\mathbb{E}}_{s_t,a_t,s_{t'},a_{t'},s_{t'+1}\sim\pi_\theta}[f(t,t')]
\end{aligned}
$$

Reward-to-go PG - Proof

We can break the expectation using the chain rule of probability:

$$\mathop{\mathbb{E}}_{s_t,a_t,s_{t'},a_{t'},s_{t'+1}\sim\pi_\theta}[f(t,t')] = \mathop{\mathbb{E}}_{s_{t'},a_{t'},s_{t'+1}\sim\pi_\theta}\left[\mathop{\mathbb{E}}_{s_t,a_t\sim\pi_\theta}[f(t,t') \mid s_{t'}, a_{t'}, s_{t'+1}]\right].$$

Substituting $f(t,t')$ in the equation above and noting that the reward term is constant w.r.t. the inner expectation:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{s_{t'},a_{t'},s_{t'+1}\sim\pi_\theta}\left[\mathop{\mathbb{E}}_{s_t,a_t\sim\pi_\theta}[f(t,t') \mid s_{t'}, a_{t'}, s_{t'+1}]\right] \\
&= \mathop{\mathbb{E}}_{s_{t'},a_{t'},s_{t'+1}\sim\pi_\theta}\left[\mathop{\mathbb{E}}_{s_t,a_t\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\mathcal{R}(s_{t'+1} \mid s_{t'}, a_{t'}) \mid s_{t'}, a_{t'}, s_{t'+1}]\right] \\
&= \mathop{\mathbb{E}}_{s_{t'},a_{t'},s_{t'+1}\sim\pi_\theta}\left[\mathcal{R}(s_{t'+1} \mid s_{t'}, a_{t'}) \mathop{\mathbb{E}}_{s_t,a_t\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \mid s_{t'}, a_{t'}, s_{t'+1}]\right].
\end{aligned}$$

Reward-to-go PG - Proof

Finally, since $t' < t$ we can break the inner expectation further using the chain rule of probability

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{s_t, a_t \sim \pi_\theta} \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \mid s_{t'}, a_{t'}, s_{t'+1} \right] \\
&= \mathop{\mathbb{E}}_{s_t \sim \pi_\theta} \left[\mathop{\mathbb{E}}_{a_t \sim \pi_\theta} \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \mid s_t \right] \middle| s_{t'}, a_{t'}, s_{t'+1} \right] \\
&= 0
\end{aligned}
$$

Which follows since for any* probability distribution,

$$
\int_x P_\theta(x) \nabla_\theta \log P_\theta(x) = \int_x \nabla_\theta P_\theta(x) = \nabla_\theta \underbrace{\int_x P_\theta(x)}_{=1} = 0.
$$

Second-Order Taylor Expansion of KL

$$\begin{aligned} D_{\mathrm{KL}}\left(\theta' \parallel \theta\right) \approx\ & D_{\mathrm{KL}}\left(\theta' \parallel \theta'\right) + \left(\nabla_\theta D_{\mathrm{KL}}\left(\theta' \parallel \theta\right)|_{\theta=\theta'}\right)^\intercal (\theta - \theta') \\ & + \frac{1}{2}(\theta - \theta')^\intercal H (\theta - \theta'), \end{aligned}$$

where

$$\begin{aligned} \nabla_\theta D_{\mathrm{KL}}\left(\theta' \parallel \theta\right)|_{\theta=\theta'} &= -\int_x p_{\theta'}(x) \frac{\nabla_\theta p_\theta(x)|_{\theta=\theta'}}{p_{\theta'}(x)} \\ &= -\int_x p_{\theta'}(x) \nabla_{\theta'} \log p_{\theta'}(x) dx = -\int_x \nabla_{\theta'} p_{\theta'}(x) dx = 0 \end{aligned}$$

and

$$\begin{aligned} H &= \nabla_\theta^2 D_{\mathrm{KL}}\left(\theta' \parallel \theta\right)|_{\theta=\theta'} \\ &= -\int_x p_{\theta'}(x) \nabla_{\theta'}^2 \log p_{\theta'}(x) dx \\ &= -\mathbb{E}_{p_{\theta'}}[\nabla_\theta^2 \log p_{\theta'}(x)], \end{aligned}$$

J. Achiam, D. Held, A. Tamar, and P. Abbeel.
Constrained policy optimization.
In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 22–31. PMLR, 2017.

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

S. M. Kakade.
A natural policy gradient.
In *Advances in neural information processing systems*, pages 1531–1538, 2002.

H. Mao, M. Alizadeh, I. Menache, and S. Kandula.
Resource management with deep reinforcement learning.
In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pages 50–56. ACM, 2016.

J. Schulman, N. Heess, T. Weber, and P. Abbeel.
Gradient estimation using stochastic computation graphs.
In *NIPS*, pages 3528–3536, 2015.

J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel.
High-dimensional continuous control using generalized advantage estimation.
In *ICLR*, 2016.

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