Data-driven and Learning-based Control

Policy optimization

Erica Salvato

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Table of Contents

What we know so far?

1 A brief recap

We introduced **Reinforcement Learning**:

- as a discrete-time, stochastic or deterministic optimal control problem
- where system's dynamical model is unknown $\rightarrow$ **Model-free control**
- and the only essential requirements are the **reward function** and data acquired directly from the real system.

$$
\begin{aligned}
\pi^* &= \arg\max_{\pi} \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \boldsymbol{h}\left(\boldsymbol{x}^{(k)}, \pi\left(\boldsymbol{x}^{(k)}\right), \boldsymbol{x}^{(k+1)}\right)\right] \\
&= \arg\max_{\pi} \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \boldsymbol{r}^{(k+1)}\right]
\end{aligned}
$$

What we know so far?

We observed that we can classify Reinforcement Learning approaches in:

- **Value-function methods:**
  The policy is implicitly defined via $V(x^{(k)})$ or $Q(x^{(k)}, u^{(k)})$ directly relying on Bellman's equations

What we know so far?

1 A brief recap

We observed that we can classify Reinforcement Learning approaches in:

- **Value-function methods:**
  The policy is implicitly defined via $V(x^{(k)})$ or $Q(x^{(k)}, u^{(k)})$ directly relying on Bellman's equations

- **Policy optimization methods:**
  The policy is a parameterized function whose weights are learned to maximize the expected cumulative discounted reward

What we know so far?

We observed that we can classify Reinforcement Learning approaches in:

- **Value-function methods:**
  The policy is implicitly defined via $V(x^{(k)})$ or $Q(x^{(k)}, u^{(k)})$ directly relying on Bellman's equations $\rightarrow$ **Critic**

- **Policy optimization methods:**
  The policy is a parameterized function whose weights are learned to maximize the expected cumulative discounted reward $\rightarrow$ **Actor**

What we know so far?

We observed that we can classify Reinforcement Learning approaches in:

- **Value-function methods:**
  The policy is implicitly defined via $V(x^{(k)})$ or $Q(x^{(k)}, u^{(k)})$ directly relying on Bellman's equations $\rightarrow$ **Critic**
- **Policy optimization methods:**
  The policy is a parameterized function whose weights are learned to maximize the expected cumulative discounted reward $\rightarrow$ **Actor**
- **Actor-critic methods:**
  Merging the two ideas by guiding the actor's learning based on the critic's estimated return

What we know so far?

1 A brief recap

We studied and applied **Value-function methods**.

We observed that they can be classified:

- Depending on the evaluation procedure in:
  - On-policy algorithms
    - SARSA
  - Off-policy algorithms
    - Q-learning
- Depending on the state space representation
  - Tabular
    - SARSA
    - Q-learning
  - Linear function approximation
    - SARSA
    - Q-learning
  - Non-linear function approximation
    - Deep Q-learning

Here, we assumed to work with a **discrete compact action set** $\mathcal{U}$.

What we know so far?

We studied and applied **Value-function methods**.

We observed that they can be classified:

- Depending on the evaluation procedure in:
  - On-policy algorithms
    - SARSA
  - Off-policy algorithms
    - Q-learning
- Depending on the state space representation
  - Tabular
    - SARSA
    - Q-learning
  - Linear function approximation
    - SARSA
    - Q-learning
  - Non-linear function approximation
    - Deep Q-learning

Here, we assumed to work with a **discrete compact action set** $\mathcal{U}$. $\rightarrow$ **What if we can't?**

Table of Contents

Policy-Based methods

Policy-based methods aim to learn directly parametrized policy

$$\pi_\theta\left(u|x\right) = Pr\left(u^{(k)}|x^{(k)}, \theta\right)$$

- Advantages:
  - Allow to work also with high-dimensional or continuous action spaces
  - Learn stochastic policies
- Disadvantages:
  - Typically converge to a local rather than global optimum
  - Evaluating a policy is typically inefficient and high variance

Epsilon-Greedy vs. Stochastic Policies

- **Epsilon-Greedy Policy**
  - Exploit (choose best action) with probability $1 - \epsilon$
  - Explore (choose random action) with probability ϵ
  - Stochastic component: when to explore and exploration
  - Deterministic component: exploitation

N.B. Even when it explores it chooses equally among all actions, i.e., the probability of choosing the worst-appearing action is equal to that of choosing the next-to-best action.

Epsilon-Greedy vs. Stochastic Policies

- **Epsilon-Greedy Policy**
  - Exploit (choose best action) with probability $1 - \epsilon$
  - Explore (choose random action) with probability ϵ
  - Stochastic component: when to explore and exploration
  - Deterministic component: exploitation

N.B. Even when it explores it chooses equally among all actions, i.e., the probability of choosing the worst-appearing action is equal to that of choosing the next-to-best action.

- **Stochastic Policy**
  - Probabilistic action selection
  - Probability distribution over actions

- Assume working with an episodic approach
- In an episode the RL controller interacts with the system and collects a trajectory

$$\tau = \left(x^{(0)}, u^{(0)}, x^{(1)}, u^{(1)}, \dots, x^{(H)}, u^{(H)}\right)$$

- We define the trajectory distribution

$$p(\tau) = p\left(x^{(0)}\right) \prod_{k=0}^{H-1} \pi_\theta\left(u^{(k)} | x^{(k)}\right) T\left(x^{(k+1)} | x^{(k)}, u^{(k)}\right)$$

- The RL objective can be then expressed as an expectation under the trajectory distribution

$$\pi_\theta^* = \arg\max_{\pi_\theta} \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{i=0}^{H-1} \gamma^i r^{(i+1)}\right]$$

The goal is to find the optimal parameter vector $\theta^* \in \mathbb{R}^t$ such that

$$\theta^* = \arg\max_{\pi_\theta} \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{i=0}^{H-1} \gamma^i r^{(i+1)} \right]$$

To simplify the notation assume $\gamma^i r^{(i+1)} = R\left(x^{(i)}, u^{(i)}\right)$, therefore:

$$\theta^* = \arg\max_{\pi_\theta} \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{i=0}^{H-1} R\left(x^{(i)}, u^{(i)}\right) \right]$$

Policy-Based methods

The goal is to find the optimal parameter vector $\theta^* \in \mathbb{R}^t$ such that

$$\theta^* = \arg\max_{\pi_\theta} \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{i=0}^{H-1} \gamma^i r^{(i+1)} \right]$$

To simplify the notation assume $\gamma^i r^{(i+1)} = R\left(x^{(i)}, u^{(i)}\right)$, therefore:

$$\theta^* = \arg\max_{\pi_\theta} \mathbb{E}_{\tau \sim p(\tau)} \underbrace{\left[\sum_{i=0}^{H-1} R\left(x^{(i)}, u^{(i)}\right) \right]}_{J(\theta)}$$

Therefore we have to optimize $J(\theta)$ with respect to θ

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{i=0}^{H-1} R\left(x^{(i)}, u^{(i)}\right) \right]$$

Therefore we have to optimize $J(\theta)$ with respect to θ

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{i=0}^{H-1} R\left(x^{(i)}, u^{(i)}\right) \right]$$

The procedure is as follows:

1. Estimate the gradient $\nabla_\theta J(\theta)$

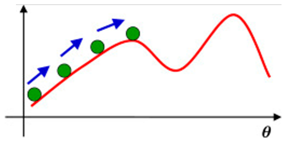
Therefore we have to optimize $J(\theta)$ with respect to θ

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau)}\left[\sum_{i=0}^{H-1} R\left(x^{(i)}, u^{(i)}\right)\right]$$

The procedure is as follows:

1. Estimate the gradient $\nabla_\theta J(\theta)$
2. Cast the learning process as approximate gradient ascent on $J(\theta)$

$$\theta = \theta + \alpha \nabla_\theta J(\theta) = \theta + \alpha \begin{bmatrix} \frac{\partial J(\theta)}{\theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\theta_t} \end{bmatrix}$$



where α is a step-size parameter

Direct policy gradient

How can we compute $\nabla_\theta J(\theta)$?

- to easy the notation define $r(\tau) = \sum_{i=0}^{H-1} R\left(x^{(i)}, u^{(i)}\right)$. therefore

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau)}\left[r(\tau)\right] = \int p_\theta(\tau)\, r(\tau)\, d\tau$$

- the gradient therefore can be written as follows:

$$\nabla_\theta J(\theta) = \int \nabla_\theta p_\theta(\tau)\, r(\tau)\, d\tau$$

Direct policy gradient

How can we compute $\nabla_\theta J(\theta)$?

- to easy the notation define $r(\tau) = \sum_{i=0}^{H-1} R\left(x^{(i)}, u^{(i)}\right)$. therefore

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau)}\left[r(\tau)\right] = \int p_\theta(\tau)\, r(\tau)\, d\tau$$

- the gradient therefore can be written as follows:

$$\nabla_\theta J(\theta) = \int p_\theta(\tau) \frac{\nabla_\theta p_\theta(\tau)}{p_\theta(\tau)} r(\tau)\, d\tau$$

How can we compute $\nabla_\theta J(\theta)$?

- to easy the notation define $r(\tau) = \sum_{i=0}^{H-1} R\left(x^{(i)}, u^{(i)}\right)$. therefore

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau)}\left[r(\tau)\right] = \int p_\theta(\tau)\, r(\tau)\, d\tau$$

- the gradient therefore can be written as follows:

$$\nabla_\theta J(\theta) = \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) r(\tau)\, d\tau$$

How can we compute $\nabla_\theta J(\theta)$?

- to easy the notation define $r(\tau) = \sum_{i=0}^{H-1} R\left(x^{(i)}, u^{(i)}\right)$. therefore

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau)}\left[r(\tau)\right] = \int p_\theta(\tau)\, r(\tau)\, d\tau$$

- the gradient therefore can be written as follows:

$$\begin{aligned}\nabla_\theta J(\theta) &= \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) r(\tau)\, d\tau \\ &= \mathbb{E}_{\tau \sim p(\tau)}\left[\nabla_\theta \log p_\theta(\tau)\, r(\tau)\right]\end{aligned}$$

Direct policy gradient

Recall that

$$p(\tau) = p\left(x^{(0)}\right) \prod_{k=0}^{H-1} \pi_\theta\left(u^{(k)} | x^{(k)}\right) T\left(x^{(k+1)} | x^{(k)}, u^{(k)}\right)$$

Then

$$\log p(\tau) = \log p\left(x^{(0)}\right) \sum_{k=0}^{H-1} \log \pi_\theta\left(u^{(k)} | x^{(k)}\right) \log T\left(x^{(k+1)} | x^{(k)}, u^{(k)}\right)$$

and therefore:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p(\tau)}\left[\nabla_\theta \log p_\theta(\tau) r(\tau)\right] = \mathbb{E}_{\tau \sim p(\tau)}\left[\nabla_\theta \left[\sum_{k=0}^{H-1} \log \pi_\theta\left(u^{(k)} | x^{(k)}\right)\right] r(\tau)\right]$$

Direct policy gradient

Finally by substituting $r(\tau) = \sum_{i=0}^{H-1} R\left(x^{(i)}, u^{(i)}\right)$ we obtain

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{k=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u^{(k)} | x^{(k)} \right) \left(\sum_{i=0}^{H-1} R\left(x^{(i)}, u^{(i)} \right) \right) \right]$$

where everything inside the expectation is known.

Direct policy gradient

Finally by substituting $r(\tau) = \sum_{i=0}^{H-1} R\left(x^{(i)}, u^{(i)}\right)$ we obtain

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{k=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u^{(k)} | x^{(k)}\right) \left(\sum_{i=0}^{H-1} R\left(x^{(i)}, u^{(i)}\right) \right) \right]$$

where everything inside the expectation is known. Now, supposing to sample N trajectories we can write the expectation as follows:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{j=1}^{N} \left[\sum_{k=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u_j^{(k)} | x_j^{(k)}\right) \left(\sum_{i=0}^{H-1} R\left(x_j^{(i)}, u_j^{(i)}\right) \right) \right]$$

Direct policy gradient

2 Policy Optimization

Finally by substituting $r(\tau) = \sum_{i=0}^{H-1} R\left(x^{(i)}, u^{(i)}\right)$ we obtain

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p(\tau)}\left[\sum_{k=0}^{H-1} \nabla_\theta \log \pi_\theta\left(u^{(k)} | x^{(k)}\right)\left(\sum_{i=0}^{H-1} R\left(x^{(i)}, u^{(i)}\right)\right)\right]$$

where everything inside the expectation is known. Now, supposing to sample N trajectories we can write the expectation as follows:

$$\nabla_\theta J(\theta) \approx \underbrace{\frac{1}{N}\sum_{j=1}^{N}\left[\sum_{k=0}^{H-1} \nabla_\theta \log \pi_\theta\left(u_j^{(k)} | x_j^{(k)}\right)\right.}_{\text{maximum likelyhood estimation}}\left.\left(\sum_{i=0}^{H-1} R\left(x_j^{(i)}, u_j^{(i)}\right)\right)\right]$$

Direct policy gradient

2 Policy Optimization

Finally by substituting $r(\tau) = \sum_{i=0}^{H-1} R\left(x^{(i)}, u^{(i)}\right)$ we obtain

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p(\tau)}\left[\sum_{k=0}^{H-1} \nabla_\theta \log \pi_\theta\left(u^{(k)} | x^{(k)}\right)\left(\sum_{i=0}^{H-1} R\left(x^{(i)}, u^{(i)}\right)\right)\right]$$

where everything inside the expectation is known. Now, supposing to sample N trajectories we can write the expectation as follows:

$$\nabla_\theta J(\theta) \approx \underbrace{\underbrace{\frac{1}{N}\sum_{j=1}^{N}\left[\sum_{k=0}^{H-1} \nabla_\theta \log \pi_\theta\left(u_j^{(k)} | x_j^{(k)}\right)\right.}_{\text{maximum likelihood estimation}}\left.\left(\sum_{i=0}^{H-1} R\left(x_j^{(i)}, u_j^{(i)}\right)\right)\right]}_{\text{weighted maximum likelihood estimation}}$$

Softmax Policy

The softmax policy is a stochastic policy that selects a control input $u^{(k)}$ according to:

$$Pr\left(u^{(k)}|x^{(k)},\theta\right)=\frac{e^{\phi\left(x^{(k)},u^{(k)}\right)^\top\theta}}{\sum_i e^{\phi\left(x^{(k)},u_i\right)^\top\theta}}$$

where:

- $\phi\left(x^{(k)},u^{(k)}\right)^\top\theta$ is the approximated action-value function.

It is used, again, when actions belong to a discrete and compact set.

Gaussian Policy

The Gaussian policy is a stochastic policy that models the probability distribution over actions using a Gaussian (normal) distribution:

$$Pr\left(\boldsymbol{u}^{(k)}|\boldsymbol{x}^{(k)},\theta\right)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(\boldsymbol{u}^{(k)}-\mu)^2}{2\sigma^2}\right)$$

where:

- $\mu=\phi\left(\boldsymbol{x}^{(k)}\right)^\top\theta$ is the approximated value function.
- σ is the standard deviation

It is used, instead, in the case of continuous action space.

REINFORCE algorithm

1. Initialize $\pi_{\theta_l} = \pi_{\theta_0}$
2. Sample N trajectories τ_i, $i = 1, \ldots, N$ by running π_{θ_l} on the environment
3. Evaluate the policy gradient

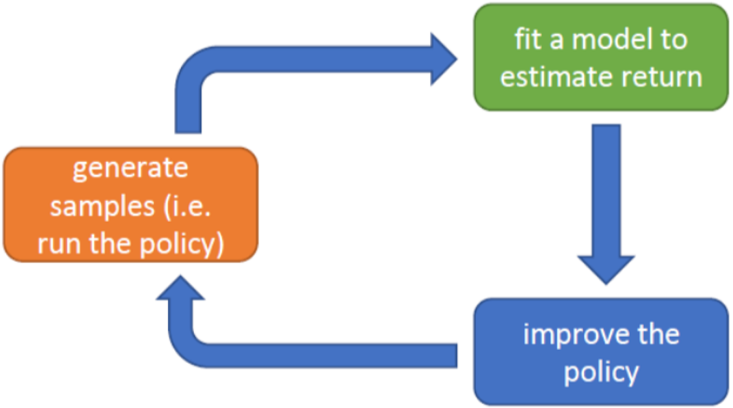
$$\nabla_{\theta_l} J\left(\theta_l\right) \approx \frac{1}{N} \sum_{j=1}^{N} \left[\sum_{k=0}^{H-1} \nabla_{\theta_l} \log \pi_{\theta_l}\left(u_j^{(k)} | x_j^{(k)}\right) \left(\sum_{i=0}^{H-1} R\left(x_j^{(i)}, u_j^{(i)}\right)\right)\right]$$

4. Take a gradient step update

$$\theta_{l+1} = \theta_l + \alpha \nabla_{\theta_l} J\left(\theta_l\right)$$

5. Repeat from step 2.

REINFORCE algorithm



Challenges of Policy Gradient Methods

Consider the policy gradient evaluation formula:

$$\nabla_{\theta_l} J(\theta_l) \approx \frac{1}{N} \sum_{j=1}^{N} \left[\sum_{k=1}^{H-1} \nabla_{\theta_l} \log \pi_{\theta_l} \left(u_j^{(k)} | x_j^{(k)} \right) \left(\sum_{i=0}^{H-1} R\left(x_j^{(i)}, u_j^{(i)} \right) \right) \right]$$

- It requires sampling entire trajectories before each gradient update

Challenges of Policy Gradient Methods

Consider the policy gradient evaluation formula:

$$\nabla_{\theta_l} J(\theta_l) \approx \frac{1}{N} \sum_{j=1}^{N} \left[\sum_{k=1}^{H-1} \nabla_{\theta_l} \log \pi_{\theta_l} \left(u_j^{(k)} | x_j^{(k)} \right) \left(\sum_{i=0}^{H-1} R\left(x_j^{(i)}, u_j^{(i)} \right) \right) \right]$$

- It requires sampling entire trajectories before each gradient update $\rightarrow$ **extensive sampling**

Challenges of Policy Gradient Methods

Consider the policy gradient evaluation formula:

$$\nabla_{\theta_l} J\left(\theta_l\right) \approx \frac{1}{N} \sum_{j=1}^{N} \left[\sum_{k=1}^{H-1} \nabla_{\theta_l} \log \pi_{\theta_l} \left(u_j^{(k)} | x_j^{(k)} \right) \left(\sum_{i=0}^{H-1} R\left(x_j^{(i)}, u_j^{(i)} \right) \right) \right]$$

- It requires sampling entire trajectories before each gradient update $\rightarrow$ **extensive sampling**
- Sampling multiple trajectories from an untrained policy leads to highly variable behaviors

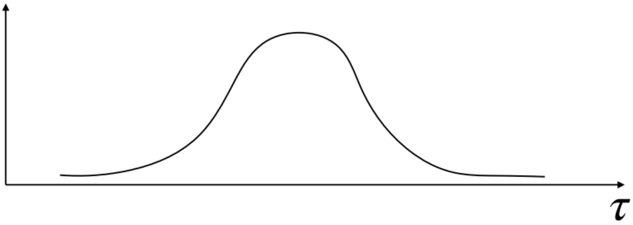
Challenges of Policy Gradient Methods

Consider the policy gradient evaluation formula:

$$\nabla_{\theta_l} J\left(\theta_l\right) \approx \frac{1}{N} \sum_{j=1}^{N} \left[\sum_{k=1}^{H-1} \nabla_{\theta_l} \log \pi_{\theta_l} \left(u_j^{(k)} | x_j^{(k)} \right) \left(\sum_{i=0}^{H-1} R\left(x_j^{(i)}, u_j^{(i)} \right) \right) \right]$$

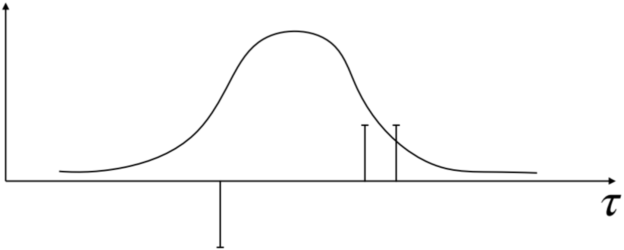
- It requires sampling entire trajectories before each gradient update $\rightarrow$ **extensive sampling**
- Sampling multiple trajectories from an untrained policy leads to highly variable behaviors $\rightarrow$ **high variance**

High Variance Policy Gradient Methods



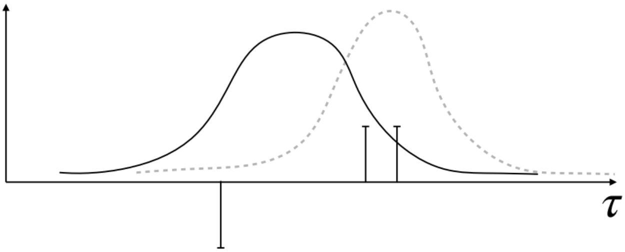
- Depending on the sample, the policy gradient can vary wildly
- This negatively affects learning: worse performance, slower convergence

High Variance Policy Gradient Methods



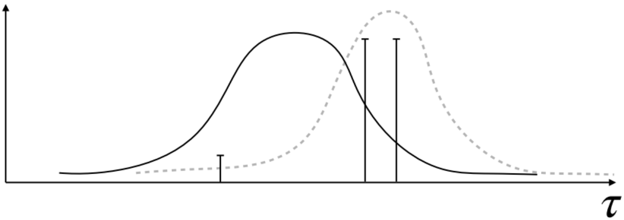
- Depending on the sample, the policy gradient can vary wildly
- This negatively affects learning: worse performance, slower convergence

High Variance Policy Gradient Methods



- Depending on the sample, the policy gradient can vary wildly
- This negatively affects learning: worse performance, slower convergence

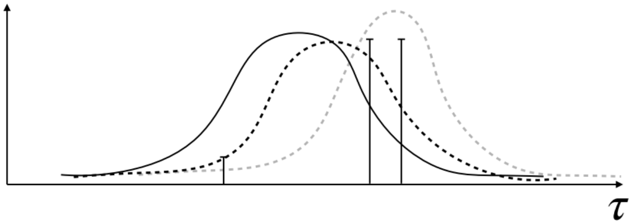
High Variance Policy Gradient Methods



- Depending on the sample, the policy gradient can vary wildly
- This negatively affects learning: worse performance, slower convergence

High Variance Policy Gradient Methods

2 Policy Optimization



- Depending on the sample, the policy gradient can vary wildly
- This negatively affects learning: worse performance, slower convergence

1. A first simple approach to reduce the variance entails using causality:

policy at time l cannot affect reward at time $i < l$

$$\nabla_{\theta_l} J\left(\theta_l\right) \approx \frac{1}{N} \sum_{j=1}^{N} \left[\sum_{k=1}^{H-1} \nabla_{\theta_l} \log \pi_{\theta_l} \left(u_j^{(k)} | x_j^{(k)} \right) \left(\sum_{i=k}^{H-1} R\left(x_j^{(i)}, u_j^{(i)} \right) \right) \right]$$

2. A second (more important) approach to reduce the variance introduces the concept of **baseline**

$$\nabla_{\theta_l} J\left(\theta_l\right) \approx \frac{1}{N} \sum_{j=1}^{N} \left[\sum_{k=1}^{H-1} \nabla_{\theta_l} \log \pi_{\theta_l} \left(x_j^{(k)}, u_j^{(k)} \right) \left(\sum_{i=1}^{H-1} R \left(x_j^{(i)}, u_j^{(i)} \right) - b \right) \right]$$

Intuitively it allows to “center” our returns, such that:

- behavior better than average gets increased
- behavior worse than average gets decreased

Notice that adding the baseline does not change the value of the expected gradient

$$
\begin{aligned}
\mathbb{E}_{\tau\sim p(\tau)}\left[\nabla_\theta \log p_\theta\left(\tau\right) b\right] &= \int p_\theta\left(\tau\right) \nabla_\theta \log p_\theta\left(\tau\right) b d\tau = \int \nabla_\theta p_\theta\left(\tau\right) b d\tau \\
&= b\nabla_\theta \int p_\theta\left(\tau\right) d\tau = b\nabla_\theta 1 = 0
\end{aligned}
$$

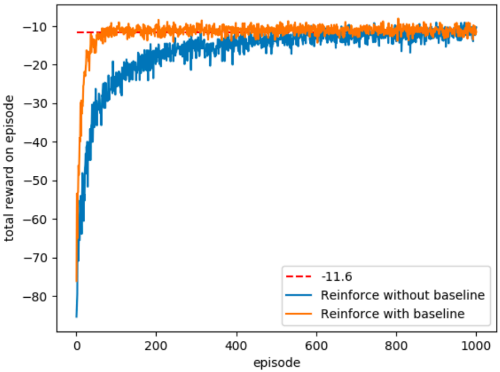
thus making our estimate of the gradient (with baseline) unbiased in expectation.

An extremely effective choice of baseline is the average return:

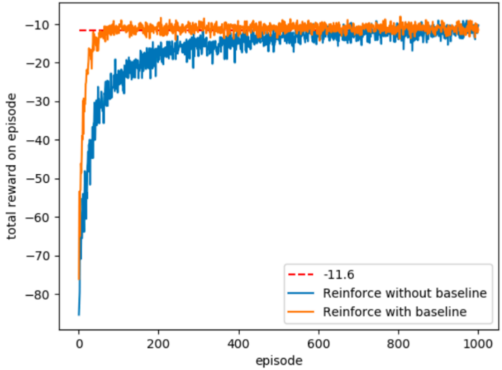
$$
\frac{1}{N}\sum_{i=1}^{N} r\left(\tau\right)
$$

Policy Gradient with and without baseline

2 Policy Optimization



Policy Gradient with and without baseline
2 Policy Optimization



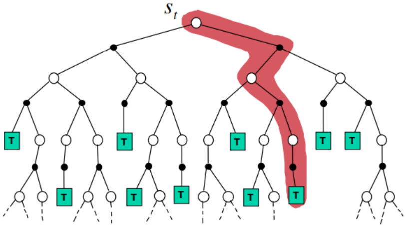
Is the policy gradient on-policy or off-policy?

Sample inefficiency of Policy Gradient Methods

Requiring a complete trajectory to compute

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{k=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u^{(k)} | x^{(k)} \right) \left(\sum_{i=0}^{H-1} R\left(x^{(i)}, u^{(i)} \right) \right) \right]$$

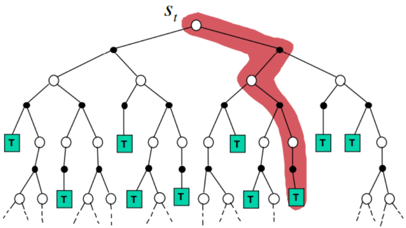
indicates our reliance on a Monte Carlo procedure once more.



Sample inefficiency of Policy Gradient Methods

Requiring a complete trajectory to compute

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{k=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u^{(k)} | x^{(k)} \right) \left(\sum_{i=0}^{H-1} R\left(x^{(i)}, u^{(i)} \right) \right) \right]$$



indicates our reliance on a Monte Carlo procedure once more.

The following main drawbacks arise:

- It requires a large number of random samples to achieve accurate results $\rightarrow$ **high computational costs**
- As the dimensionality of the problem increases, the number of samples required for accurate estimates grows exponentially $\rightarrow$ **curse of dimensionality**

Reduce Extensive Sampling of Policy Gradient Methods

A solution might be to substitute $\sum_{i=k}^{H-1} R\left(x^{(i)}, u^{(i)}\right)$ with the estimate action value function $Q\left(x^{(k)}, u^{(k)}\right)$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p(\tau)}\left[\sum_{k=0}^{H-1} \nabla_\theta \log \pi_\theta\left(u^{(k)} | x^{(k)}\right) Q\left(x^{(k)}, u^{(k)}\right)\right]$$

Hence, by applying what we already studied we can use a **Critic** to estimate the

action-value function:

$$Q_w\left(x^{(k)}, u^{(k)}\right) \approx Q\left(x^{(k)}, u^{(k)}\right)$$

In this way, the Critic update can be performed in a TD learning fashion.

Reduce Extensive Sampling of Policy Gradient Methods

A solution might be to substitute $\sum_{i=k}^{H-1} R\left(x^{(i)}, u^{(i)}\right)$ with the estimate action value function $Q\left(x^{(k)}, u^{(k)}\right)$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p(\tau)}\left[\sum_{k=0}^{H-1} \nabla_\theta \log \pi_\theta\left(u^{(k)} | x^{(k)}\right) Q\left(x^{(k)}, u^{(k)}\right)\right]$$

Hence, by applying what we already studied we can use a **Critic** to estimate the

action-value function:

$$Q_w\left(x^{(k)}, u^{(k)}\right) \approx Q\left(x^{(k)}, u^{(k)}\right)$$

In this way, the Critic update can be performed in a TD learning fashion.

$$\downarrow$$

Actor Critic

Table of Contents

Actor-critic algorithms maintain two sets of parameters:

- **Critic** updates action-value function parameters w
- **Actor** updates policy parameters θ in the direction suggested by critic

thus following an approximate policy gradient

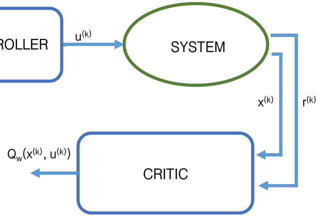
$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{k=0}^{H-1} \nabla_\theta \log \pi_\theta \left(u^{(k)} | x^{(k)} \right) Q_w \left(x^{(k)}, u^{(k)} \right) \right]$$

The Critic is performing nothing more than policy evaluation:

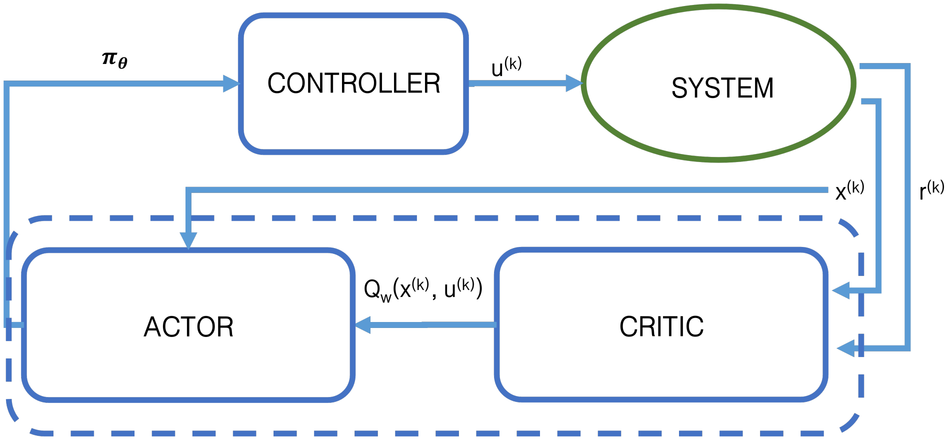
Given θ how good is policy π_θ?

We already know that this operation can be performed:

- via Monte-Carlo policy evaluation
- via Temporal-Difference learning



Actor Critic Scheme



Actor Critic Algorithm with SARSA critic

Initialization. θ, w choose $\phi(x,u)$ and π_θ

Repeat (for each episode)

- Set $x^{(0)}$
- Select an action $u^{(k)}$ with π_θ
- Repeat for each step of the episode until the terminal condition is met
  - Perform the action $u^{(k)}$, observe $x^{(k+1)}$ and $r^{(k+1)}$
  - Select an action $u^{(k+1)}$ with π_θ
  - $Q\left(x^{(k)}, u^{(k)}\right) = w^\top \phi\left(x^{(k)}, u^{(k)}\right)$ and $Q\left(x^{(k+1)}, u^{(k+1)}\right) = w^\top \phi\left(x^{(k+1)}, u^{(k+1)}\right)$
  - $\theta = \theta + \beta \nabla_\theta \log \pi_\theta\left(u^{(k)} | x^{(k)}\right) Q_w\left(x^{(k)}, u^{(k)}\right)$
  - $w = w + \alpha\left[r^{(k+1)} + \gamma Q\left(x^{(k+1)}, u^{(k+1)}\right) - Q\left(x^{(k)}, u^{(k)}\right)\right] \nabla_w Q\left(x^{(k)}, u^{(k)}\right)$
  - Update $x^{(k)} = x^{(k+1)}$, $u^{(k)} = u^{(k+1)}$

Approximated Policy Gradient

The use of approximating functions to calculate the Critic leads to the achievement of approximations of even $\nabla_\theta J(\theta)$.

- What if the chosen features are not convenient for goal achievement?
- What if the chosen features are completely wrong?

Therefore, we need to carefully choose value function approximation.

There exists a theorem that allows us to provide some guarantees.

Compatible Function Approximation Theorem

If the following two conditions are satisfied:

1. Value function approximator is compatible to the policy

$$\nabla_w Q_w(x, u) = \nabla_\theta \log \pi_\theta(u|x)$$

2. Value function parameters w minimise the mean-squared error

$$\mathbb{E}_{\pi_\theta}\left[\left(Q_{\pi_\theta}(x, u) - Q_w(x, u)\right)^2\right]$$

Then the policy gradient is exact:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(u|x)\, Q_w(x, u)\right]$$

Proof of Compatible Function Approximation Theorem

If w is such that the mean-squared error is minimized, then its gradient with respect to w must be 0:

$$\nabla_w \mathbb{E}_{\pi_\theta}\left[\left(Q_{\pi_\theta}(x,u) - Q_w(x,u)\right)^2\right] = 0$$

$$\mathbb{E}_{\pi_\theta}\left[\left(Q_{\pi_\theta}(x,u) - Q_w(x,u)\right)\nabla_w Q_w(x,u)\right] = 0$$

Imposing $\nabla_w Q_w(x,u) = \nabla_\theta \log \pi_\theta(u|x)$

$$\mathbb{E}_{\pi_\theta}\left[\left(Q_{\pi_\theta}(x,u) - Q_w(x,u)\right)\nabla_\theta \log \pi_\theta(u|x)\right] = 0$$

$$\mathbb{E}_{\pi_\theta}\left[Q_{\pi_\theta}(x,u)\nabla_\theta \log \pi_\theta(u|x)\right] = \mathbb{E}_{\pi_\theta}\left[Q_w(x,u)\nabla_\theta \log \pi_\theta(u|x)\right]$$

Therefore can be substituted in the policy gradient thus obtaining

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(u|x)\, Q_w(x,u)\right]$$

$\square$

Reduce Variance of Actor Critic Methods

Also in this case, in order to reduce the variance we can introduce a baseline

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(u|x)\left(Q_w(x,u) - b\right)\right]$$

A good baseline, in this case, is the value function $b = V_{\pi_\theta}(x)$.

Reduce Variance of Actor Critic Methods

Also in this case, in order to reduce the variance we can introduce a baseline

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(u|x)\left(Q_w(x,u) - b\right)\right]$$

A good baseline, in this case, is the value function $b = V_{\pi_\theta}(x)$.

Then by defining the advantage function

$$A_{\pi_\theta}(x,u) = Q_{\pi_\theta}(x,u) - V_{\pi_\theta}(x)$$

we can rewrite the policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(u|x)\, A_{\pi_\theta}(x,u)\right]$$

Therefore, we need to estimate the advantage function.

1. A way can consist of estimating both $V_{\pi_\theta}(x)$ and $Q_{\pi_\theta}(x, u)$ using two different function approximators with two parameter vectors v and w.

$$V_v(x) \approx V_{\pi_\theta}(x)$$

$$Q_w(x, u) \approx Q_{\pi_\theta}(x, u)$$

$$A(x, u) = Q_w(x, u) - V_v(x)$$

And updating both value functions by TD learning for example

2. A second approach can consist in observing that

$$Q_{\pi_\theta}\left(x^{(k)}, u^{(k)}\right) = r^{(k+1)} + \mathbb{E}_{\pi_\theta}\left[V\left(x^{(k+1)}\right)\right]$$

Therefore

$$A\left(x^{(k)}, u^{(k)}\right) = Q_{\pi_\theta}\left(x^{(k)}, u^{(k)}\right) - V_{\pi_\theta}\left(x^{(k)}\right) = r^{(k+1)} + V_{\pi_\theta}\left(x^{(k+1)}\right) - V_{\pi_\theta}\left(x^{(k)}\right)$$

Estimating the Advantage Function

2. A second approach can consist in observing that

$$Q_{\pi_\theta}\left(x^{(k)}, u^{(k)}\right) = r^{(k+1)} + \mathbb{E}_{\pi_\theta}\left[V\left(x^{(k+1)}\right)\right]$$

Therefore

$$A\left(x^{(k)}, u^{(k)}\right) = Q_{\pi_\theta}\left(x^{(k)}, u^{(k)}\right) - V_{\pi_\theta}\left(x^{(k)}\right) = \underbrace{r^{(k+1)} + V_{\pi_\theta}\left(x^{(k+1)}\right) - V_{\pi_\theta}\left(x^{(k)}\right)}_{\delta_{\pi_\theta}^{(k+1)}}$$

2. A second approach can consist in observing that

$$Q_{\pi_\theta}\left(x^{(k)}, u^{(k)}\right) = r^{(k+1)} + \mathbb{E}_{\pi_\theta}\left[V\left(x^{(k+1)}\right)\right]$$

Therefore

$$A\left(x^{(k)}, u^{(k)}\right) = Q_{\pi_\theta}\left(x^{(k)}, u^{(k)}\right) - V_{\pi_\theta}\left(x^{(k)}\right) = \underbrace{r^{(k+1)} + V_{\pi_\theta}\left(x^{(k+1)}\right) - V_{\pi_\theta}\left(x^{(k)}\right)}_{\delta_{\pi_\theta}^{(k+1)}}$$

Therefore we can use the TD error to compute the policy gradient

$$\nabla_\theta J\left(\theta\right) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta\left(u|x\right)\delta_{\pi_\theta}^{(k+1)}\right]$$

thus requiring only one set of parameters (those used to approximate V_{π_θ})

The policy gradient has many equivalent forms

$$\nabla_{\theta_l} J(\theta_l) \approx \frac{1}{N}\sum_{j=1}^{N}\left[\sum_{k=0}^{H-1} \nabla_{\theta_l} \log \pi_{\theta_l}\left(u_j^{(k)}|x_j^{(k)}\right)\left(\sum_{i=0}^{H-1} R\left(x_j^{(i)}, u_j^{(i)}\right)\right)\right]$$ REINFORCE

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p(\tau)}\left[\sum_{k=0}^{H-1} \nabla_\theta \log \pi_\theta\left(u^{(k)}|x^{(k)}\right) Q_w\left(x^{(k)}, u^{(k)}\right)\right]$$ Actor Critic with SARSA critic

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p(\tau)}\left[\sum_{k=0}^{H-1} \nabla_\theta \log \pi_\theta\left(u^{(k)}|x^{(k)}\right) A_w\left(x^{(k)}, u^{(k)}\right)\right]$$ Advantage Actor Critic

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p(\tau)}\left[\sum_{k=0}^{H-1} \nabla_\theta \log \pi_\theta\left(u^{(k)}|x^{(k)}\right) \delta\right]$$ TD Actor Critic

- Each one leading to a stochastic gradient ascent algorithm
- Critic uses policy evaluation (e.g. MC or TD learning) to estimate Q, A or V

Table of Contents

Course summary

4 Conlcusion

a) We started introducing the **optimal control problems** in the general form, highlighting the roles of **controllers** and **dynamical systems**.

b) We observed how to **solve optimal problems** in cases where the dynamical system **model is known**

- Markov Decision Process (MDP)
- Linear Quadratic Regulator (LQR)

c) We observed how optimal control problems involving **non-linear dynamics** can be solved **with LQR**

- Iterative Linear Quadratic Regulator (iLQR)

d) We then moved to the so-called **model-free approaches**, where a model of the dynamical system is not needed

- Monte Carlo (MC)
- Temporal Difference learning (TD)
- Reinforcement Learning (RL)

e) We studied how to apply **RL in different settings**:

- Bellman's equation-based (or **Value function**) approaches: considering different design choices of the state space
    - Tabular
    - Linear Function approximation
    - Non-linear Function approximation (Neural networks)

Course summary

4 Conlcusion

- Approaches that learn the policy directly: allowing for different choices of the action space (**Policy Gradient**).
- Hybrid approaches: combining both ideas, thus allowing to work with all possible choices of both state and action spaces (**Actor Critic**).

Questions' time!

UNIVERSITÀ DEGLI STUDI DI TRIESTE