Robot Learning

Deep RL Tutorial

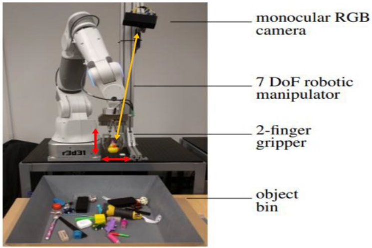
Glen Berseth

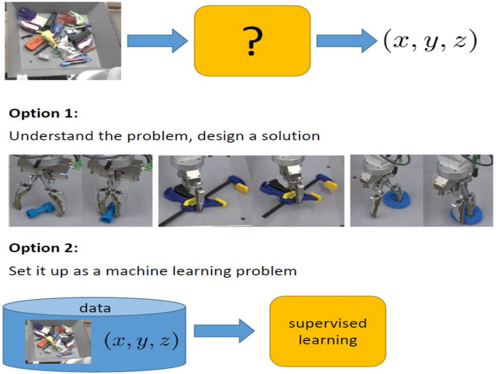
Université de Montréal

November 7, 2022

- www.fracturedplane.com
- glen.berseth@umontreal.ca
- Some material from Levine DeepRL Course

Outline





- There are many situations where traditional models are challenged - Large state spaces - Non-linear dynamics - Discontinuous contacts

What Problem is DeepRL Solving?

No feature engineering!

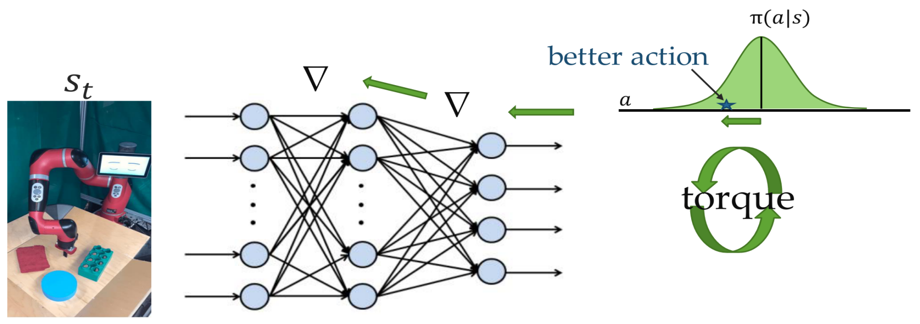


Figure: Deep Learning and Reinforcement Learning

- The perception and planning problem in a more general way.

What Problem is DeepRL Solving?

Sensor Motor Loop

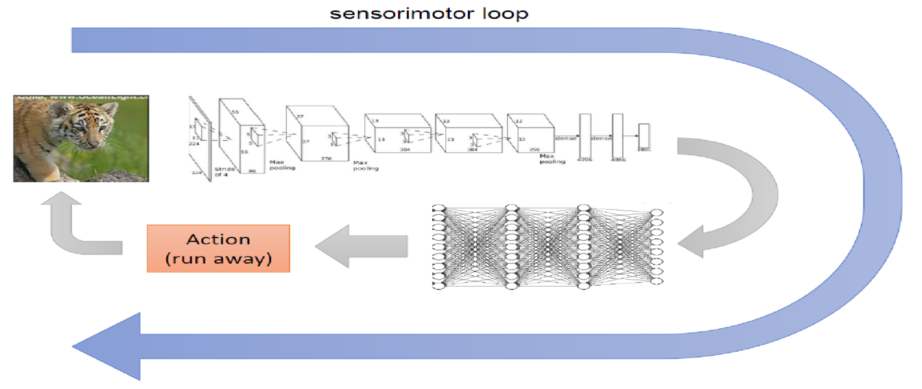


Figure: Sensory motor loop

- RL agents collect their own data to solve a task
  - No need for expert data

Supervised learning

- given $\mathcal{D} = \{x_i, y_i\}$
  - learn to predict y_i given x_i, $y \leftarrow f(x)$
- Assumptions in supervised learning
  - Data is Independant and Identically Distributed (IID)
    - This is rarely the case in the real world
  - True optimal action y is known
- Example:
  - $L(\theta) = ||f(x|\theta) - y||^2$

Supervised learning

- given $\mathcal{D} = \{x_i, y_i\}$
  - learn to predict y_i given x_i, $y \leftarrow f(x)$
- Assumptions in supervised learning
  - Data is Independant and Identically Distributed (IID)
    - ★ This is rarely the case in the real world
  - True optimal action y is known
- Example:
  - $L(\theta) = ||f(x|\theta) - y||^2$

Reinforcement Learning

- Previous outputs influence future inputs
  - Data is not IID
- Optimal action y is known
  - Instead we have a scalar reward function
- **reward** function
  - $r \leftarrow R(s, a)$
  - **weighted regression**
- Example:
  - $L(\theta) = ||f(s|\theta) - a||^2 R(s, a)$

What is Reinforcement Learning

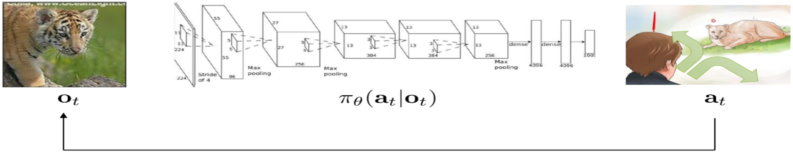


Figure: First terms

- a_t - Action
- $\mathbf{a}_t$ - Continuous action
- $\mathbf{s}_t$ - State
- $\mathbf{o}_t$ - Observation

What is Reinforcement Learning

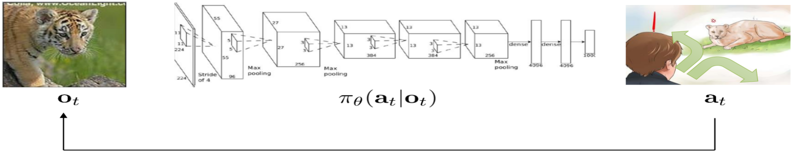


Figure: First terms

- a_t - Action
- $\mathbf{a}_t$ - Continuous action
- $\mathbf{s}_t$ - State
- $\mathbf{o}_t$ - Observation

- $\pi(\mathbf{a}_t|\mathbf{o}_t, \theta)$ policy
- $\pi(\mathbf{a}_t|\mathbf{s}_t, \theta)$ fully observed policy

What is Reinforcement Learning

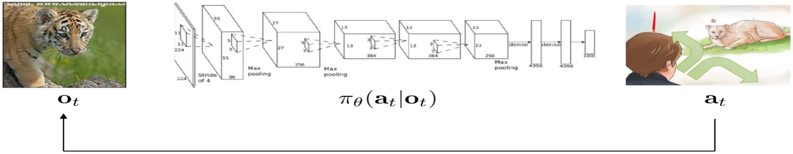


Figure: First terms

- a_t - Action
- $\mathbf{a}_t$ - Continuous action
- $\mathbf{s}_t$ - State
- $\mathbf{o}_t$ - Observation
- $\pi(\mathbf{a}_t|\mathbf{o}_t, \theta)$ policy
- $\pi(\mathbf{a}_t|\mathbf{s}_t, \theta)$ fully observed policy

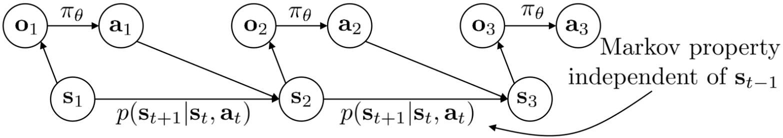


Figure: Markov property

Reinforcement Learning Objective

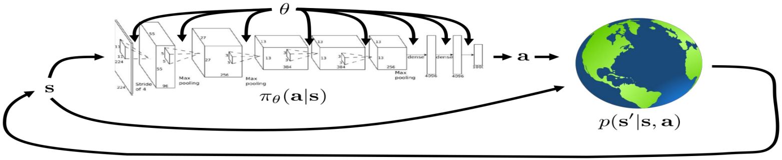


Figure: Reinforcement Learning Environment

Reinforcement Learning Objective

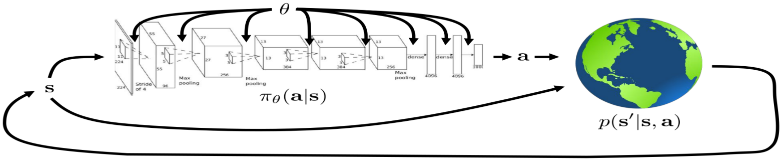


Figure: Reinforcement Learning Environment

- Distribution over trajectories $p(\tau|\theta)$ using chain rule of probability

$$\underbrace{p(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T|\theta)}_{p(\tau|\theta)} = \underbrace{p(\mathbf{s}_1)}_{\text{unknown}} \prod_{t=1}^{T} \pi(\mathbf{a}_t|\mathbf{s}_t, \theta) \underbrace{p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)}_{\text{unknown}} \tag{1}$$

Reinforcement Learning Objective

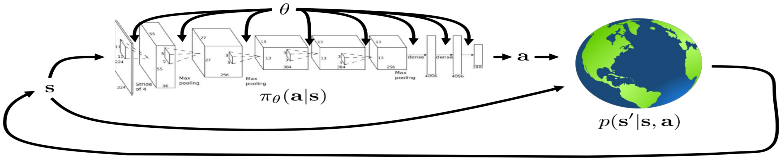


Figure: Reinforcement Learning Environment

- Distribution over trajectories $p(\tau|\theta)$ using chain rule of probability

$$\underbrace{p(\mathbf{s}_1, \mathbf{a}_1, \dots, \mathbf{s}_T, \mathbf{a}_T|\theta)}_{p(\tau|\theta)} = \underbrace{p(\mathbf{s}_1)}_{\text{unknown}} \prod_{t=1}^{T} \pi(\mathbf{a}_t|\mathbf{s}_t, \theta) \underbrace{p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)}_{\text{unknown}} \tag{1}$$

- RL objective is over this distribution

$$\arg\max_{\theta^*} \mathbb{E}_{\tau \sim p(\tau|\theta)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right] \tag{2}$$

Basic Reinforcement Learning Loop: (1) Collect Data

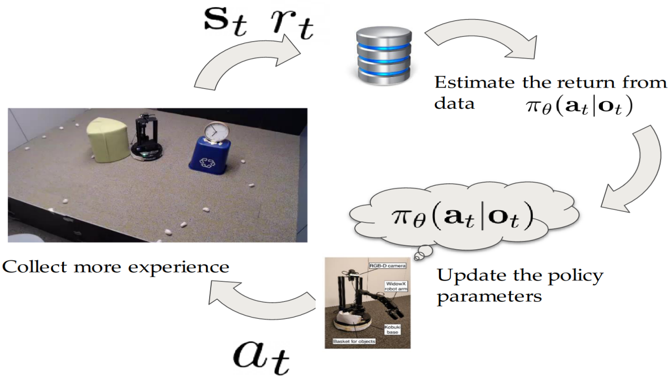


Figure: Sensory motor loop

Basic Reinforcement Learning Loop: (1) Collect Data

Collect Data

```
import gym
env = gym.make("LunarLander-v2") ## Create an instance of the control environm
observation, info = env.reset(seed=42, return_info=True) ## Reset the environm
buff = [] ## Array to store experience
for _ in range(1000):
   env.render()  ## Render the environment if desired
   action = policy(observation)  # User-defined policy function
   next_observation, reward, done, info = env.step(action) ## Take a step in t
   buff.append([observation, action, reward, next_observation])
   observation = next_observation
   if done:
      observation, info = env.reset(return_info=True) ## Reset if the robot ha

env.close()
```

Basic Reinforcement Learning Loop: (2) Estimate Return/Score

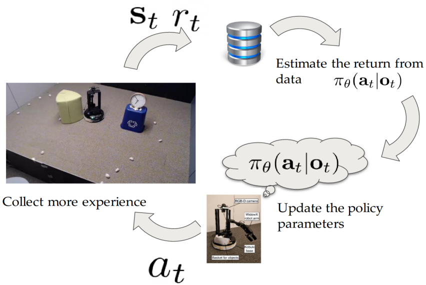


Figure: Sensory motor loop

Basic Reinforcement Learning Loop: (2) Estimate Return/Score

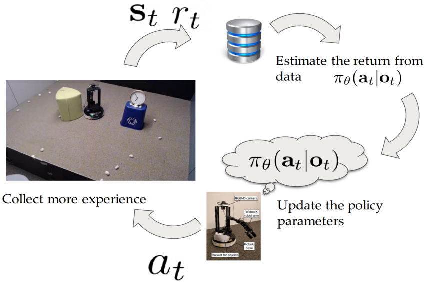


Figure: Sensory motor loop

Estimate the return for θ

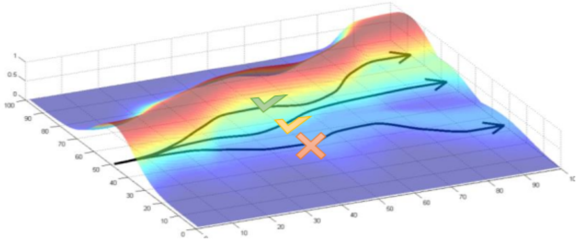


Figure: Policy Gradient

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau|\theta)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right] \tag{3}$$

Basic Reinforcement Learning Loop: (3) Update The Policy

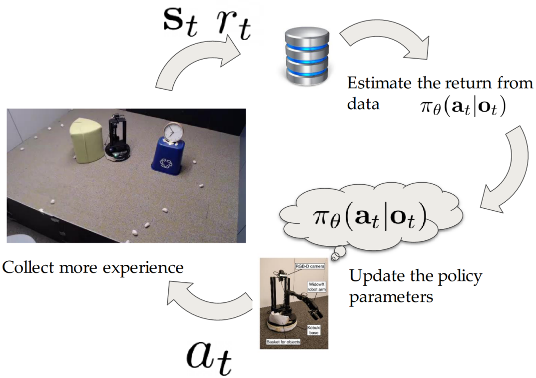


Figure: Sensory motor loop

Update the policy

Basic Reinforcement Learning Loop: (3) Update The Policy

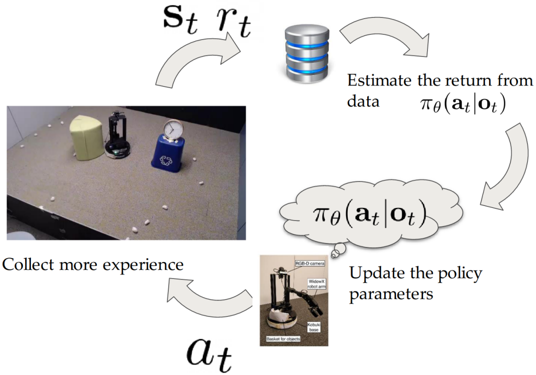


Figure: Sensory motor loop

Update the policy

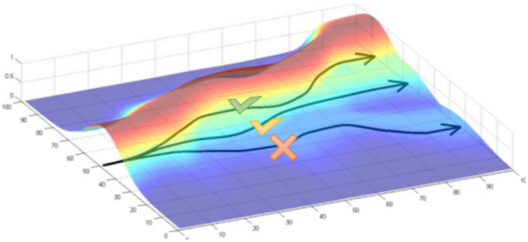


Figure: Policy Gradient

- $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
- α is the learning rate

You need to train a model

- Model-Based Reinforcement Learning (MBRL)
- Why learn a model?
  - For most problems the dynamics are unknown
  - If we have $\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$ we can plan (see last week)
- Then all we need to do is learn $\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$, that should be *easy*.

You need to train a model

- MBRL
- Why learn a model?
  - For most problems the dynamics are unknown
  - If we have $\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$ we can plan (see last week)
- Then all we need to do is learn $\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$, that should be *easy*.

Basic MBRL

1. Collect experience $< \mathbf{s}_{t+1}, \mathbf{s}_t, \mathbf{a}_t > \in \mathcal{D}_{\text{train}}$ from the environment with $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$
2. Train θ to minimize $\sum_i ||f(\mathbf{s}_t, \mathbf{a}_t, \theta) - \mathbf{s}_{t+1}||$
3. Use $f(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \theta)$ to plan high reward trajectories

(Wang *et al.*, 2018)

Model-Based Reinforcement Learning

How Well Does Basic MBRL work

- Not that well, why?

How Well Does Basic MBRL work

- Not that well, why?

Basic MBRL

1: Collect experience $< \mathbf{s}_{t+1}, \mathbf{s}_t, \mathbf{a}_t > \in \mathcal{D}_{\text{train}}$ from the environment with $\pi_{\text{rand}}(\mathbf{a}_t|\mathbf{s}_t)$
2: Train θ to minimize $\sum_i ||f(\mathbf{s}_t, \mathbf{a}_t, \theta) - \mathbf{s}_{t+1}||$
3: Use $f(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \theta)$ to plan high value trajectories

- Goal: Move higher
- But: $\pi_{\text{rand}}(\mathbf{a}_t|\mathbf{s}_t) \neq \pi(\mathbf{a}_t|\mathbf{s}_t, \theta)$

- Problem grows with model complexity

How to train a forward model

- How to reduce $\pi_{\text{rand}}(\mathbf{a}_t|\mathbf{s}_t) \neq \pi(\mathbf{a}_t|\mathbf{s}_t, \theta)$
- Ideas?

How to train a forward model

- How to reduce $\pi_{\text{rand}}(\mathbf{a}_t|\mathbf{s}_t) \neq \pi(\mathbf{a}_t|\mathbf{s}_t, \theta)$
- Ideas?
- Need more on policy data [Dagger](Ross *et al.*, 2011)

How to train a forward model

- How to reduce $\pi_{\text{rand}}(\mathbf{a}_t|\mathbf{s}_t) \neq \pi(\mathbf{a}_t|\mathbf{s}_t, \theta)$
- Ideas?
- Need more on policy data [Dagger](Ross *et al.*, 2011)

OnPolicy MBRL

1: Collect experience $< \mathbf{s}_{t+1}, \mathbf{s}_t, \mathbf{a}_t > \in \mathcal{D}_{\text{train}}$ from the environment with $\pi_{\text{rand}}(\mathbf{a}_t|\mathbf{s}_t)$
2: **while** true **do**
3: Train θ to minimize $\sum_i ||f(\mathbf{s}_t, \mathbf{a}_t, \theta) - \mathbf{s}_{t+1}||$
4: Use $f(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \theta)$ to plan high value trajectories
5: Collect experience $< \mathbf{s}_{t+1}, \mathbf{s}_t, \mathbf{a}_t > \in \mathcal{D}_{\text{train}}$from the environment with $f(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \theta)$
6: **end while**

How to train a forward model

- How to reduce $\pi_{\text{rand}}(\mathbf{a}_t|\mathbf{s}_t) \neq \pi(\mathbf{a}_t|\mathbf{s}_t, \theta)$
- Ideas?
- Need more on policy data [Dagger](Ross *et al.*, 2011)

OnPolicy MBRL

Collect experience $< \mathbf{s}_{t+1}, \mathbf{s}_t, \mathbf{a}_t > \in \mathcal{D}_{\text{train}}$ from the environment with $\pi_{\text{rand}}(\mathbf{a}_t|\mathbf{s}_t)$
while true **do**
    Train θ to minimize $\sum_i ||f(\mathbf{s}_t, \mathbf{a}_t, \theta) - \mathbf{s}_{t+1}||$
    Use $f(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \theta)$ to plan high value trajectories
    Collect experience $< \mathbf{s}_{t+1}, \mathbf{s}_t, \mathbf{a}_t > \in \mathcal{D}_{\text{train}}$from the environment with $f(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \theta)$
end while

- What is wrong with this algorithm?
    - Hint: What objective is it optimizing?

(Deisenroth and Rasmussen, 2011; Chua *et al.*, 2018; Hafner *et al.*, 2019)

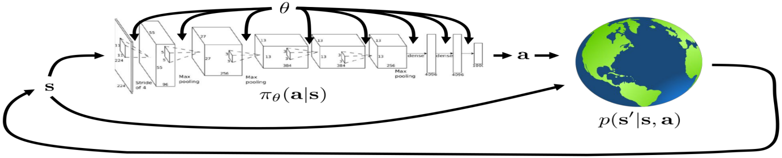


Figure: Reinforcement Learning Environment

- Distribution over trajectories $p(\tau|\theta)$ using chain rule of probability

$$\underbrace{p(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T|\theta)}_{p(\tau|\theta)} = \underbrace{p(\mathbf{s}_1)}_{\text{Unknown}} \prod_{t=1}^{T} \pi(\mathbf{a}_t|\mathbf{s}_t, \theta) \underbrace{p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)}_{\text{Now unknown}} \tag{4}$$

- RL objective is over this distribution

$$\arg\max_{\theta^*} \mathbb{E}_{\tau\sim p(\tau|\theta)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right] \tag{5}$$

- **MBRL is not optimizing for this objective.** (Joseph *et al.*, 2013; Farahmand *et al.*, 2017; Janner *et al.*, 2019; Grimm *et al.*, 2020; Lambert *et al.*, 2020; Nikishin *et al.*, 2022)

The Policy Gradient

$$\theta^* = \arg\max_{\theta} \underbrace{\mathbb{E}_{\tau \sim p(\tau|\theta)} \left[\sum_{t} r(\mathbf{s}_t, \mathbf{a}_t) \right]}_{J(\theta)} \quad (6)$$

- How can we use this?

The Policy Gradient

$$\theta^* = \arg\max_{\theta} \underbrace{\mathbb{E}_{\tau \sim p(\tau|\theta)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]}_{J(\theta)} \quad (6)$$

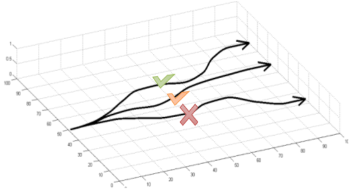


Figure: Simple policy Gradient

- How can we use this?
- Approximate with samples from the environment

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau|\theta)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right] \approx \frac{1}{N} \sum_n^N \sum_t^T r(\mathbf{s}_{n,t}, \mathbf{a}_{n,t}) \quad (7)$$

The Policy Gradient

$$\theta^* = \arg\max_\theta \underbrace{\mathbb{E}_{\tau\sim p(\tau|\theta)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]}_{J(\theta)} \quad (6)$$

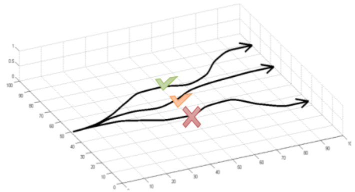


Figure: Simple policy Gradient

- How can we use this?
- Approximate with samples from the environment

$$J(\theta) = \mathbb{E}_{\tau\sim p(\tau|\theta)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right] \approx \frac{1}{N}\sum_n^N \sum_t^T r(\mathbf{s}_{n,t}, \mathbf{a}_{n,t}) \quad (7)$$

- **Unbiased** estimate of the expected value
- Simple to perform direct gradient ascent

Examples: Reinforce (Williams, 1992; Sutton *et al.*, 2000)

Basic Reinforcement Learning Loop: Update Policy

Reducing Variance: Baselines

- $\nabla_\theta J(\theta) = \frac{1}{N}\sum_{i=1}^{N} \nabla \log p(\tau) r(\tau)$
- Average reward
    - $b_t = \frac{1}{N}\sum_{i=1}^{N} r(\tau)$
    - Reweight trajectories by their average performance

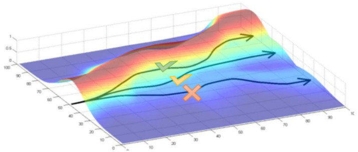


Figure: Policy Gradient

Reducing Variance: Baselines

- $\nabla_\theta J(\theta) = \frac{1}{N}\sum_{i=1}^{N} \nabla \log p(\tau) r(\tau)$
- Average reward
  - $b_t = \frac{1}{N}\sum_{i=1}^{N} r(\tau)$
  - Reweight trajectories by their average performance

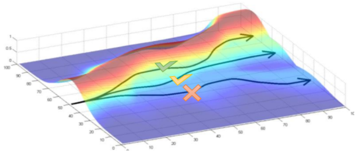


Figure: Policy Gradient

- Will this change the optimal policy?
- $\mathbb{E}[\nabla_\theta \log p(\tau|\theta) b] = \int p(\tau) \nabla_\theta \log p(\tau|\theta) b d\tau$
  - Use identity
- $\int \nabla_\theta p(\tau|\theta) b d\tau = b \nabla_\theta \int p(\tau|\theta) d\tau = b \nabla_\theta 1 = 0$
  - Same optimal policy

Load your robot model

- Create a simulated environment for the control loop
  - Or a real environment
- Create a reward function
  - Easy in simulation, often difficult in the real world

OpenAiGym API

```
env = gym.make(env_id)
env = gym.wrappers.RecordEpisodeStatistics(env)
```

DeepRL and Robotics

OpenAIGym Wrappers for Preprocesing

```
## Deep Networks like outputs in [-1,1]
env = gym.wrappers.ClipAction(env)
## Deep Networks like inputs in [-1,1]
env = gym.wrappers.NormalizeObservation(env)
env = gym.wrappers.TransformObservation(env, lambda obs: np.clip(obs, -10, 10)
## DeepRL likes rewards [-1,1]
env = gym.wrappers.NormalizeReward(env, gamma=gamma)
env = gym.wrappers.TransformReward(env, lambda reward: np.clip(reward, -10, 1
```

- This way learning rates, etc have meaning

Many RL libraries to use

- Stable Baselines: Good place to start
- cleanrl: simple implimentations of RL algorithms
- rlkit: Designed for robotics applications
- tf_agents: Based on deepmind applications
- Many others..

Many RL libraries to use

- Stable Baselines: Good place to start
- cleanrl: simple implimentations of RL algorithms
- rlkit: Designed for robotics applications
- tf_agents: Based on deepmind applications
- Many others..

- Learn how to use RL first with simple examples
  - See my class
- Then upgrade to code for real experiments.

DeepRL Tutorial

- cleanrl:
- Setup code here.
  - https://github.com/milarobotlearningcourse/cleanrl/blob/master/roble_install.md

DeepRL Tutorial

- cleanrl:
- Setup code here.
  - https://github.com/milarobotlearningcourse/cleanrl/blob/master/roble_install.md
- Fix code in ppo_continuous_action.py
  - https://github.com/milarobotlearningcourse/cleanrl/blob/master/cleanrl/ppo_continuous_a
  - look for “TODO ##”
  - Ask questions!

Scratch

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. May 2018.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472. Citeseer, 2011.

Amir-Massoud Farahmand, Andre Barreto, and Daniel Nikovski. Value-Aware Loss Function for Model-based Reinforcement Learning. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1486–1494. PMLR, 2017.

Christopher Grimm, André Barreto, Satinder Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. pages 5541–5552, November 2020.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. pages 1–19, December 2019.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. June 2019.

Joshua Joseph, Alborz Geramifard, John W Roberts, Jonathan P How, and Nicholas Roy. Reinforcement learning with misspecified model classes. In *2013 IEEE International Conference on Robotics and Automation*, pages 939–946, May 2013.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. February 2020.

Evgenii Nikishin, Romina Abachi, Rishabh Agarwal, and Pierre-Luc Bacon. Control-Oriented Model-Based reinforcement learning with implicit differentiation. *AAAI*, 36(7):7886–7894, June 2022.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063. proceedings.neurips.cc, 2000.

Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. NerveNet: Learning structured policy with graph neural networks. February 2018.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3):229–256, May 1992.