POLICY SEARCH WITH POLICY GRADIENT

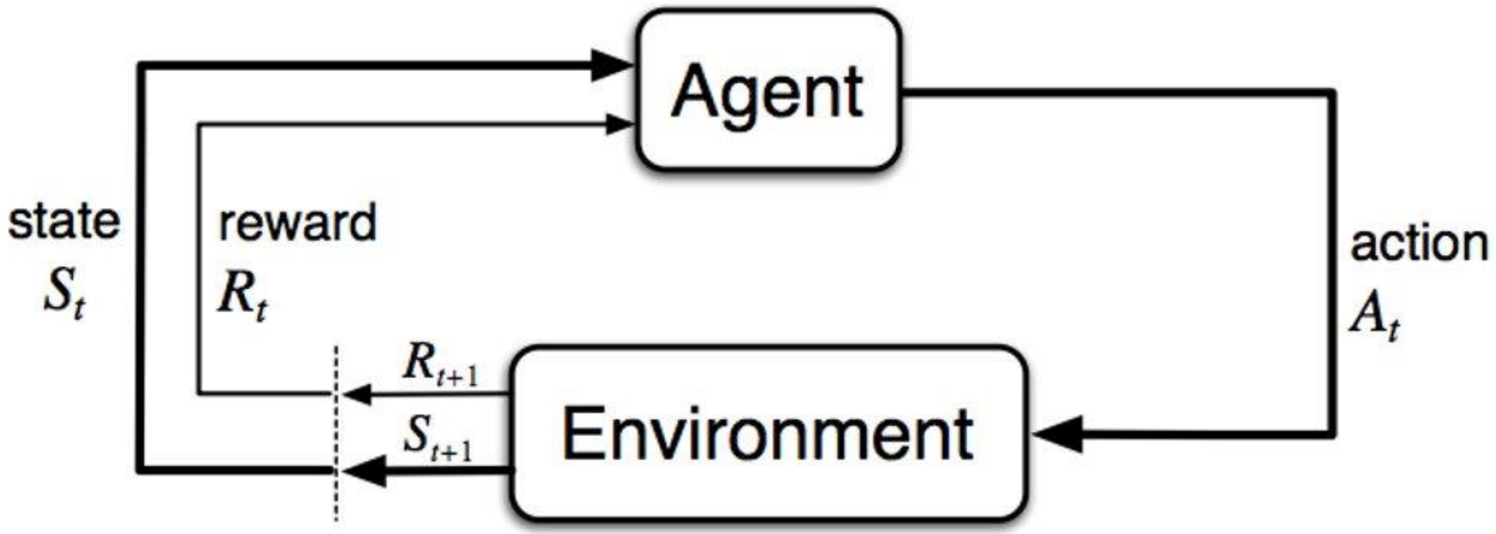
Mohammad Firouzi

CS411 Tutorial – Fall 2018

Adapted from Sergey Levine and David Silver slides

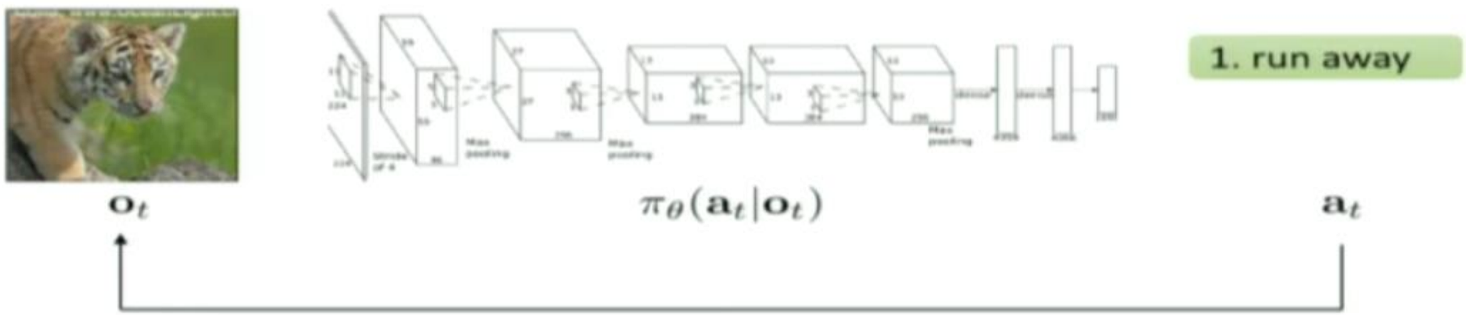
The agent-environment interface

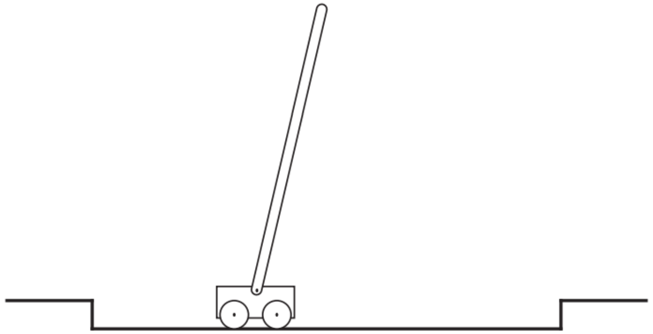
- Environment
- Agent
- State
- Action
- Reward



Examples

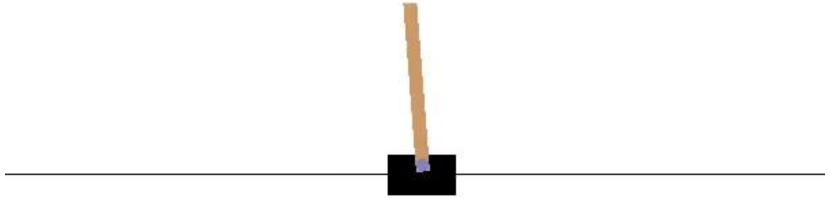
- Pick-and-place robot
- Mars robot
- Pole-balancing robot
- Supervised learning as reinforcement learning
- Atari games

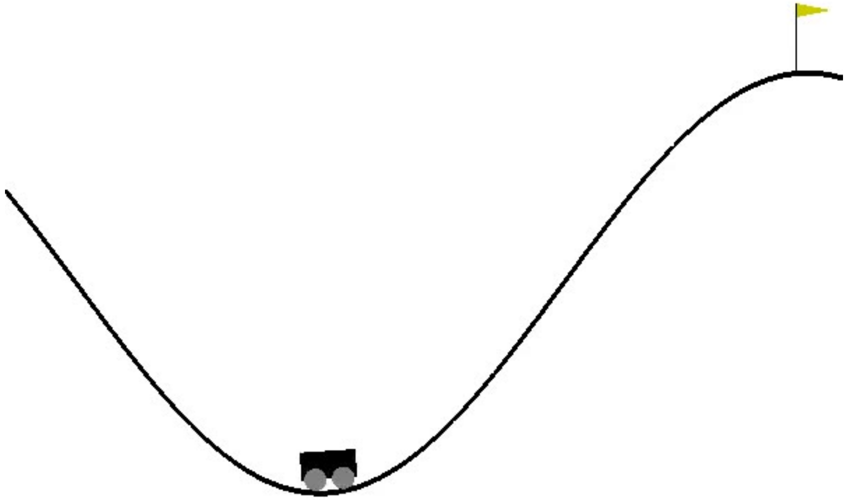


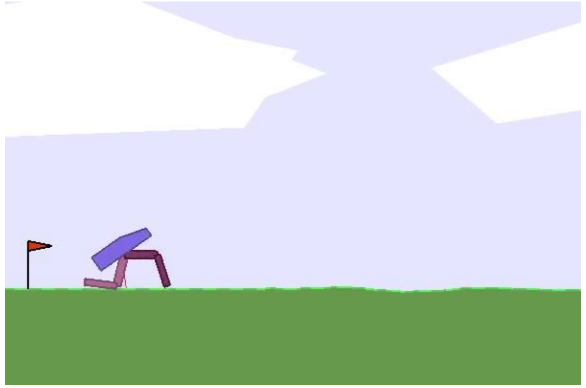


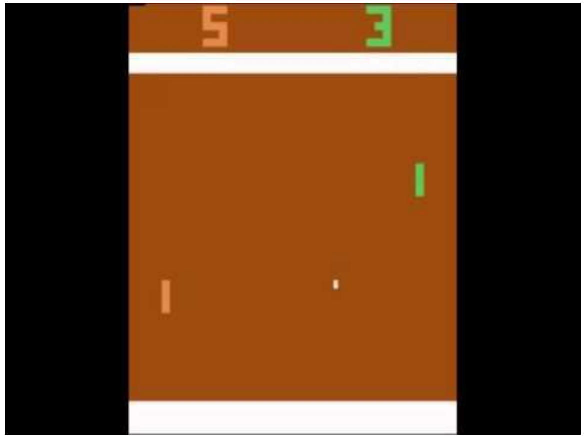
OpenAI gym environments

- CartPole
- MountainCar
- Pong
- BeamRider
- BipedalWalker
- …

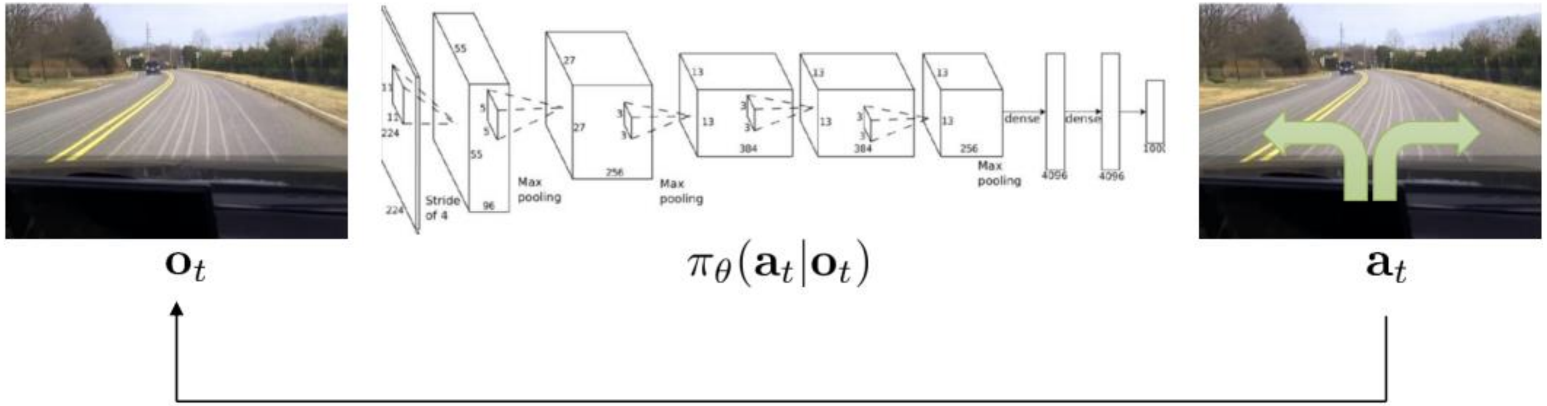






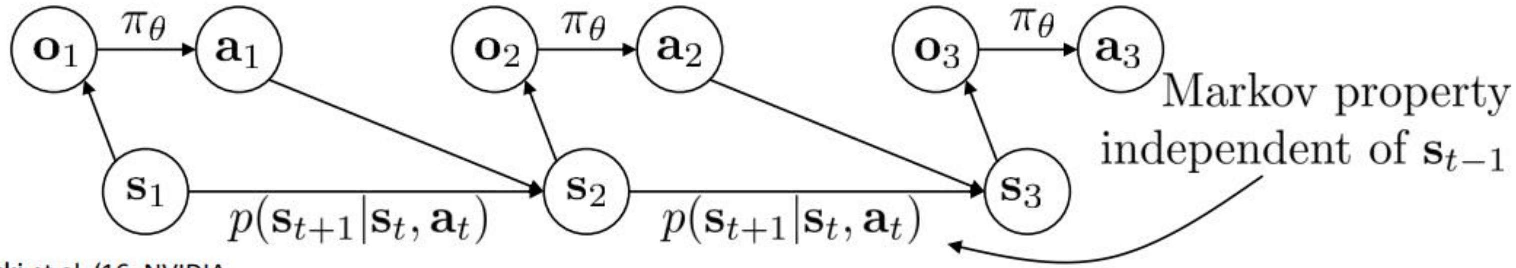


Terminology & Notation



$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action

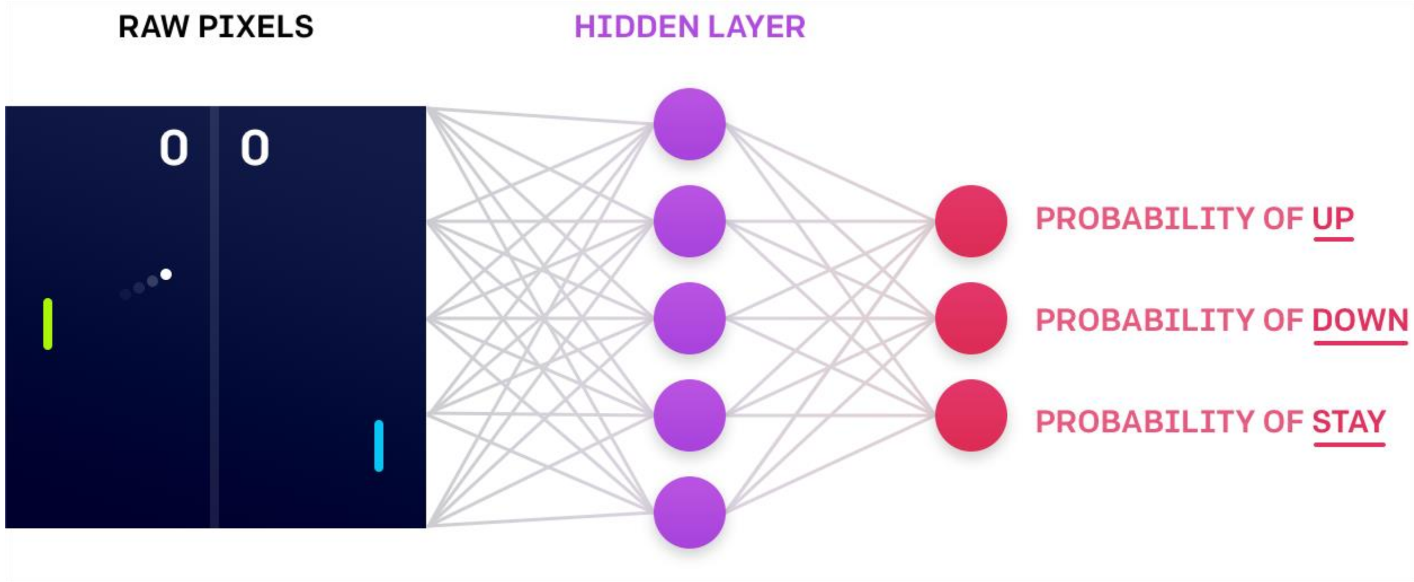
$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ – policy
$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ – policy (fully observed)



Images: Bojarski et al. '16, NVIDIA

Policy

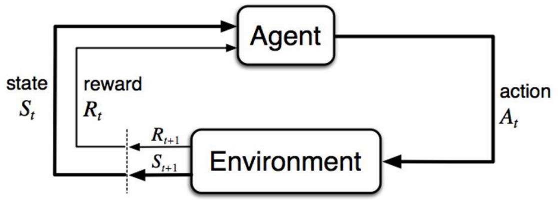
- A policy is the agent's behaviour
- It is a map from state space to action space:
  - Deterministic policy
  - Stochastic policy



Goal of reinforcement learning

- Obtain a policy that maximizes the expected rewards

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



$$\theta^\star = \arg\max_\theta E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

$$\theta^\star = \arg\max_\theta E_{(\mathbf{s},\mathbf{a}) \sim p_\theta(\mathbf{s},\mathbf{a})} [r(\mathbf{s}, \mathbf{a})]$$

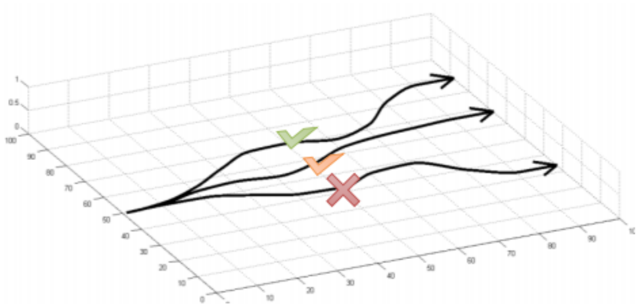
infinite horizon case

$$\theta^\star = \arg\max_\theta \sum_{t=1}^{T} E_{(\mathbf{s}_t,\mathbf{a}_t) \sim p_\theta(\mathbf{s}_t,\mathbf{a}_t)} [r(\mathbf{s}_t, \mathbf{a}_t)]$$

finite horizon case

Evaluating the objective

$$\theta^\star = \arg\max_\theta \underbrace{E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t,\mathbf{a}_t)\right]}_{J(\theta)}$$



$$J(\theta) = E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t,\mathbf{a}_t)\right] \approx \frac{1}{N}\sum_i \sum_t r(\mathbf{s}_{i,t},\mathbf{a}_{i,t})$$

sum over samples from π_θ

Approaches To Reinforcement Learning

- Policy-based RL (focus of the tutorial)
  - Search directly for the optimal policy
  - This is the policy achieving maximum future reward
- Value-based RL (will be discussed in brief)
  - Estimate the optimal value function Q(s, a)
  - This is the maximum value achievable under any policy
- Model-based RL (will be discussed in brief)
  - Build a model of the environment
  - Plan (e.g. by lookahead) using model

Value-based approach (in brief)

- A Q-value function is a prediction of future reward
  - "How much reward will I get from action a in state s?"
- Q-value function gives expected total reward
  - from state *s* and action *a*
  - under policy π
  - with discount factor γ

$$Q^\pi(s,a) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s, a\right]$$

- Q-value functions decompose into a Bellman equation

$$Q^\pi(s,a) = \mathbb{E}_{s',a'}\left[r + \gamma Q^\pi(s',a') \mid s, a\right]$$

Optimal value function

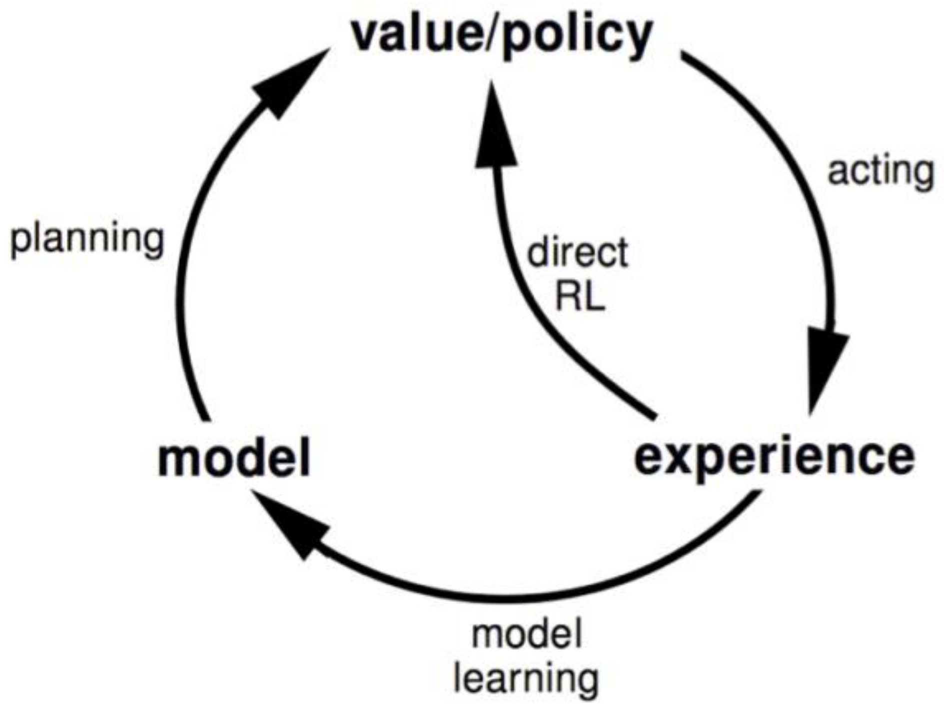
- An optimal value function is the maximum achievable value

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) = Q^{\pi^*}(s,a)$$

- Once we have optimal Q-value function we can act optimally

$$\pi^*(s) = \operatorname*{argmax}_{a} Q^*(s,a)$$

Model-based approach (in brief)



Policy-based approach

- Direct policy differentiation

$$\underbrace{\theta^\star = \arg\max_\theta E_{\tau\sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t,\mathbf{a}_t)\right]}_{J(\theta)}$$

a convenient identity

$$\pi_\theta(\tau)\nabla_\theta\log\pi_\theta(\tau) = \pi_\theta(\tau)\frac{\nabla_\theta\pi_\theta(\tau)}{\pi_\theta(\tau)} = \nabla_\theta\pi_\theta(\tau)$$

$$J(\theta) = E_{\tau\sim p_\theta(\tau)}[\underbrace{r(\tau)}_{\sum_{t=1}^{T} r(\mathbf{s}_t,\mathbf{a}_t)}] = \int \pi_\theta(\tau)r(\tau)d\tau$$

$$\nabla_\theta J(\theta) = \int \nabla_\theta\pi_\theta(\tau)r(\tau)d\tau = \int \pi_\theta(\tau)\nabla_\theta\log\pi_\theta(\tau)r(\tau)d\tau = E_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta\log\pi_\theta(\tau)r(\tau)]$$

Direct policy differentiation

$$\theta^\star = \arg\max_\theta J(\theta)$$

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}[r(\tau)]$$

$$\nabla_\theta J(\theta) = E_{\tau\sim p_\theta(\tau)}[\nabla_\theta \underline{\log \pi_\theta(\tau)} r(\tau)]$$

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

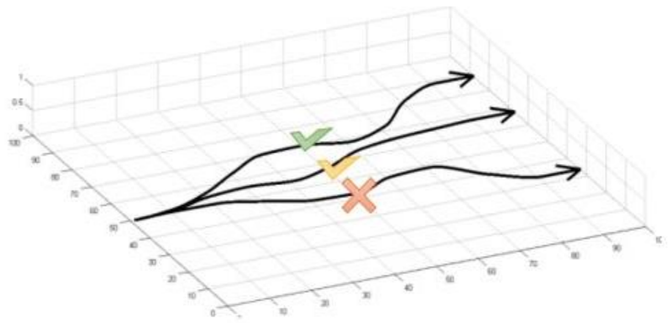
log of both sides

$$\underline{\log p_\theta(\tau) = \log p(\mathbf{s}_1) + \sum_{t=1}^{T} \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)}$$

$$\nabla_\theta \left[\cancel{\log p(\mathbf{s}_1)} + \sum_{t=1}^{T} \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \cancel{\log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)} \right]$$

$$\nabla_\theta J(\theta) = E_{\tau\sim p_\theta(\tau)}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\right)\right]$$

Evaluating the policy gradient

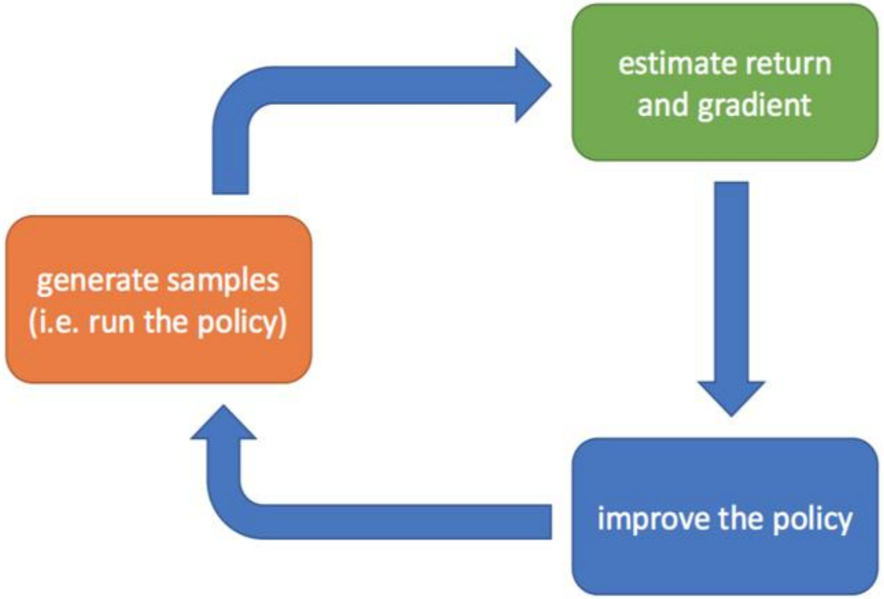


recall: $J(\theta) = E_{\tau \sim p_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right] \approx \frac{1}{N}\sum_i \sum_t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$

$$\nabla_\theta J(\theta) = E_{\tau \sim p_\theta(\tau)}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\right)\right]$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})\right)$$

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$



REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)\left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Reducing Variance

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t},\mathbf{a}_{i,t})\right)$$

Causality: policy at time t' cannot affect reward at time t when $t < t'$

$$\nabla_\theta J(\theta) \approx \tfrac{1}{N}\textstyle\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi(a_{i,t} \mid s_{i,t})\underbrace{(\textstyle\sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'}))}_{\hat{Q}_{i,t}}$$

Baselines

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau)[r(\tau) - b] \qquad b = \frac{1}{N}\sum_{i=1}^{N} r(\tau)$$

a convenient identity

$$\pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) = \nabla_\theta \pi_\theta(\tau)$$

- Are we allowed to do that?

$$E[\nabla_\theta \log \pi_\theta(\tau) b] = \int \pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) b \, d\tau = \int \nabla_\theta \pi_\theta(\tau) b \, d\tau = b\nabla_\theta \int \pi_\theta(\tau) d\tau = b\nabla_\theta 1 = 0$$

- Subtracting a baseline is unbiased in expectation!
- average reward is not the best baseline, but it’s pretty good!

Policy gradient with automatic differentiation

- Pseudocode example (with discrete actions):

```
# Given:
# actions - (N*T) x Da tensor of actions
# states - (N*T) x Ds tensor of states
# q_values – (N*T) x 1 tensor of estimated state-action values
# Build the graph:
logits = policy.predictions(states) # This should return (N*T) x Da tensor of action logits
negative_likelihoods = tf.nn.softmax_cross_entropy_with_logits(labels=actions, logits=logits)
weighted_negative_likelihoods = tf.multiply(negative_likelihoods, q_values)
loss = tf.reduce_mean(weighted_negative_likelihoods)
gradients = loss.gradients(loss, variables)
```

$$\tilde{J}(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \hat{Q}_{i,t}$$

q_values

Basics of OpenAI gym

Environment

- Has attributes that give environment specificiations (e.g. action space, observation space, etc.)
- Environment *step* function gets an action and update the environment for one step
  - It returns four values, observation (i.e. observation in the next time step), reward, done (e.g. agent died!), info (e.g. it might contain the raw probabilities behind the environment's last state change)
- Can be rendered or restart by *render*, or *reset* functions

We implement the policy gradient method for CartPole (one of the gym environments).

References

- Sergey Levine, "Policy Search", Deep learning summer school slides, https://dlrlsummerschool.ca/wp-content/uploads/2018/09/levine-policy-search-rlss-2018.pdf
- David Silver, "Deep Reinforcement Learning", ICML 2016 tutorial, https://icml.cc/2016/tutorials/deep_rl_tutorial.pdf
- Sutton, Richard S., and Andrew G. Barto. *Introduction to reinforcement learning*. Vol. 135. Cambridge: MIT press, 1998.