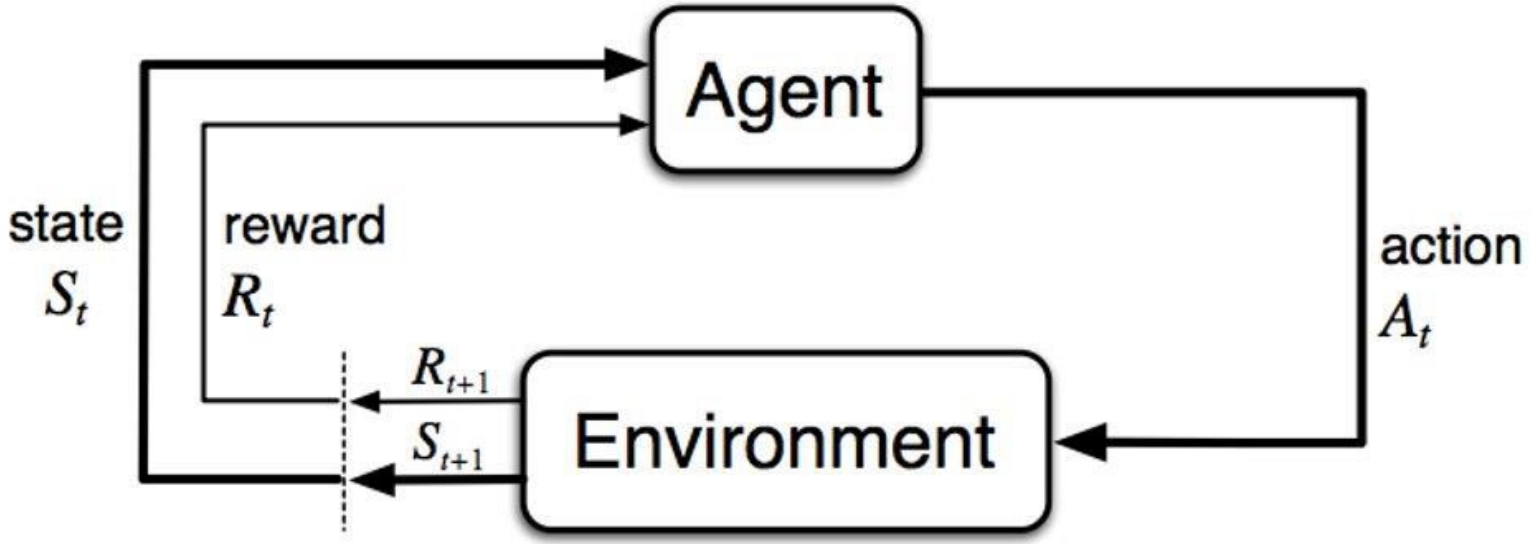
POLICY SEARCH WITH POLICY GRADIENT

Tianshi Cao

CS311 Tutorial – Fall 2019

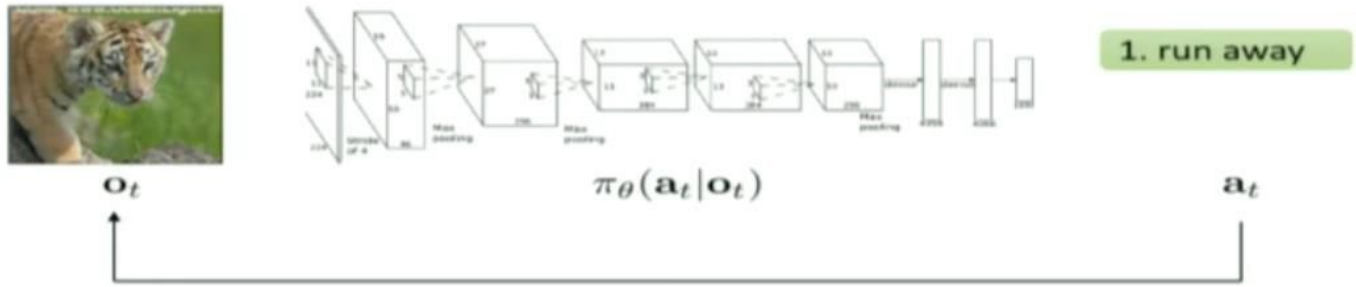
Reinforcement Learning Set-up

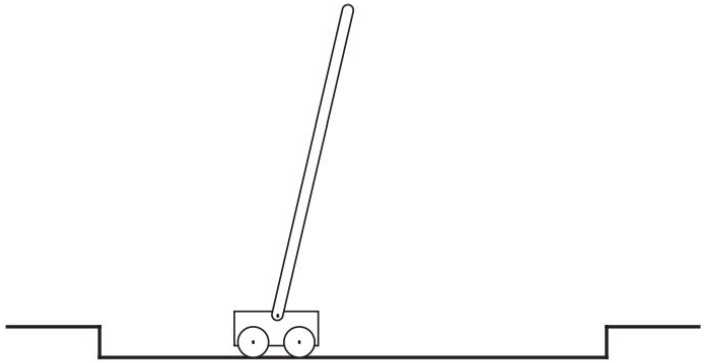
- Environment
- Agent
- State
- Action
- Reward



Examples

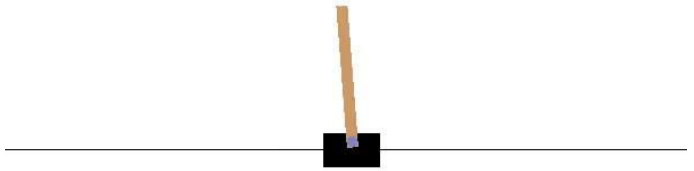
- Pick-and-place robot
- Mars robot
- Pole-balancing robot
- Imitation learning
- Atari games

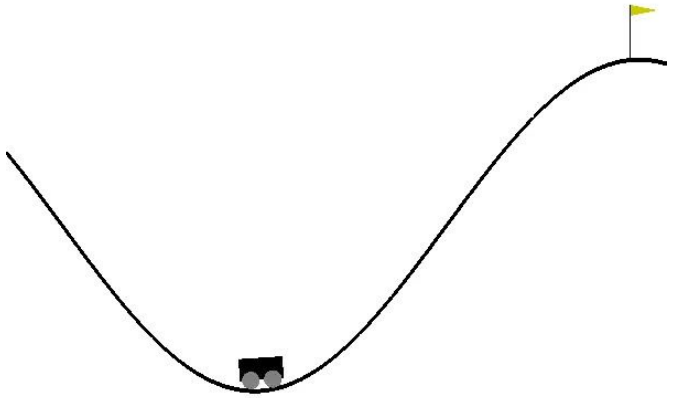


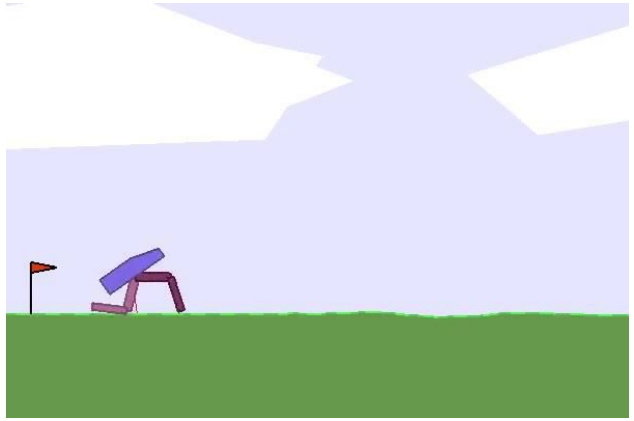


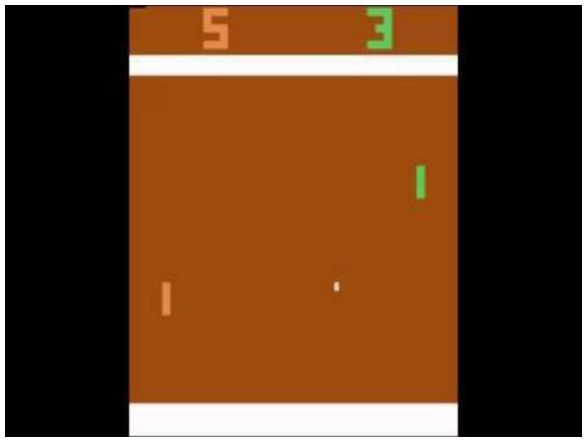
OpenAI gym environments

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









Reinforcement Learning Task

Environment $\mathbf{E} = \{\mathcal{S}, \mathcal{A}, \mathcal{F}, \rho_0\}$

$\mathcal{S}$: Space of States

$\mathcal{A}$: Space of Actions

$\mathcal{F}$: State action transition density function $\mathcal{F}(s, a, s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$

ρ_0 : Initial state distribution

Task $\mathbf{T} = \{\mathbf{E}, R_E(s), \gamma, H\}$

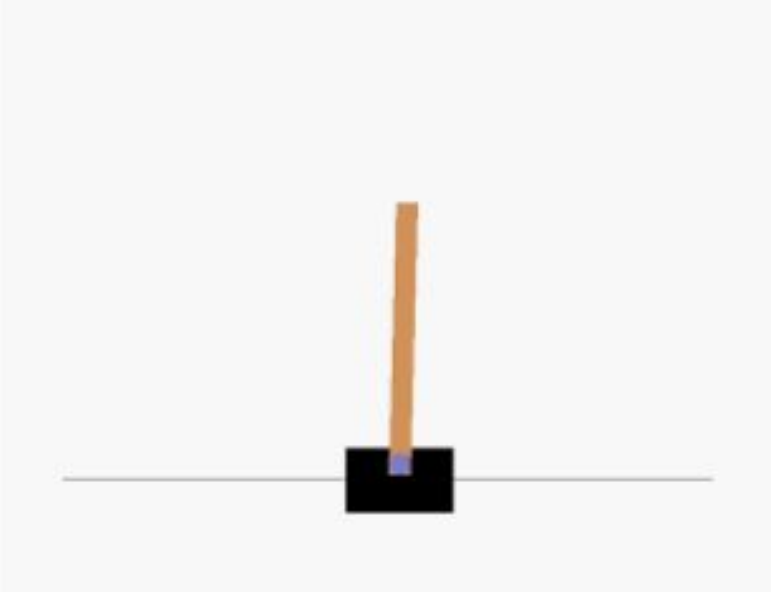
$R(s) : \mathcal{S} \mapsto \mathbb{R}$: Reward function mapping states (or state+action) to a real value

H : Horizon (how long the interaction lasts, can be infinite)

γ : Discount rate (want to get reward earlier than later)

Task Example

CartPole: balancing a pole attached to a cart by a joint



$\mathcal{S}$: position and velocity of cart and pole (4 numbers)

$\mathcal{A}$: Push cart left or right (binary)

$\mathcal{F}$: deterministic, simulated newtonian physics

ρ_0 : slightly random, cart and pole centered with near 0 velocity

$R(s)$: 1 for each frame where the pole is standing

H : user defined, generally around 100 frames

γ : generally 1 (i.e. no discount)

Agent Formulation

Agent is defined by a state-action policy: $\pi(s, a)$

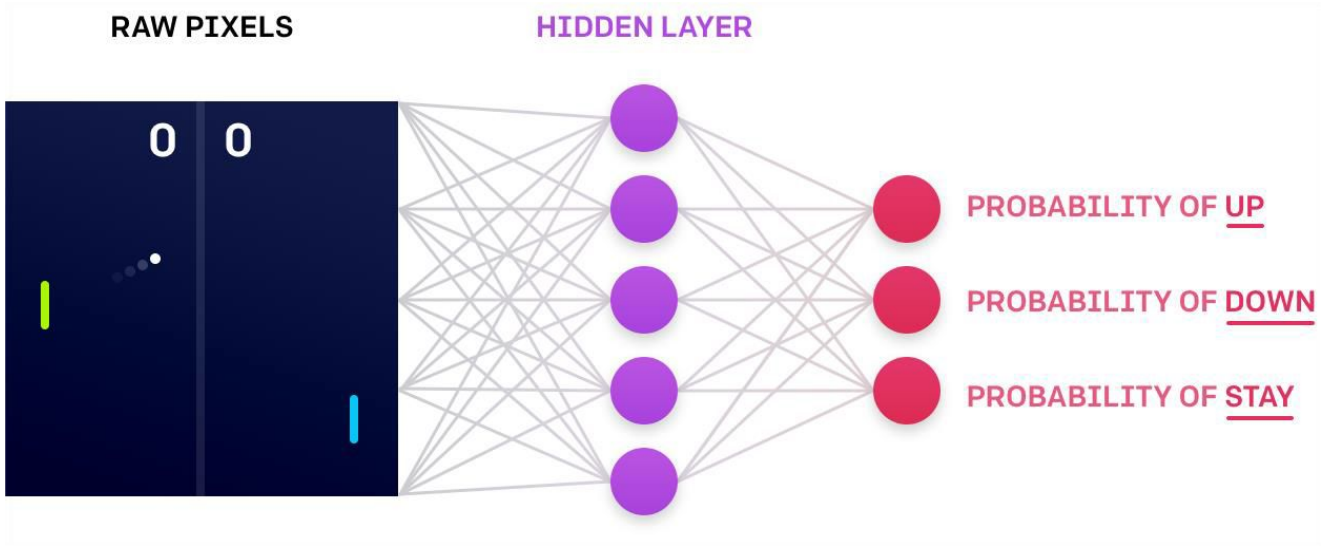
$$\mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$$

Goal of the agent:

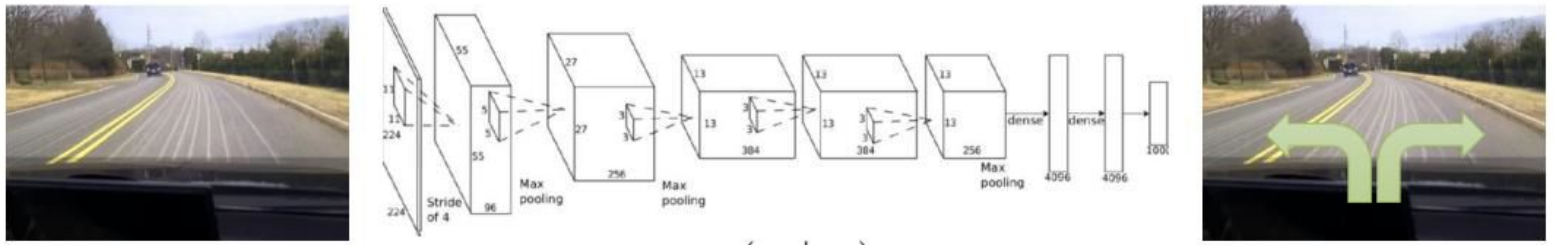
Maximize expected reward: $\mathbb{E}_{a\sim\pi, s\sim\mathcal{F}}\left[\sum_{t=1}^{H} \gamma^t R(s_t)\right]$

Agent Formulation

- π is often times parameterized by θ
- Can be either:
  - Deterministic
  - Stochastic



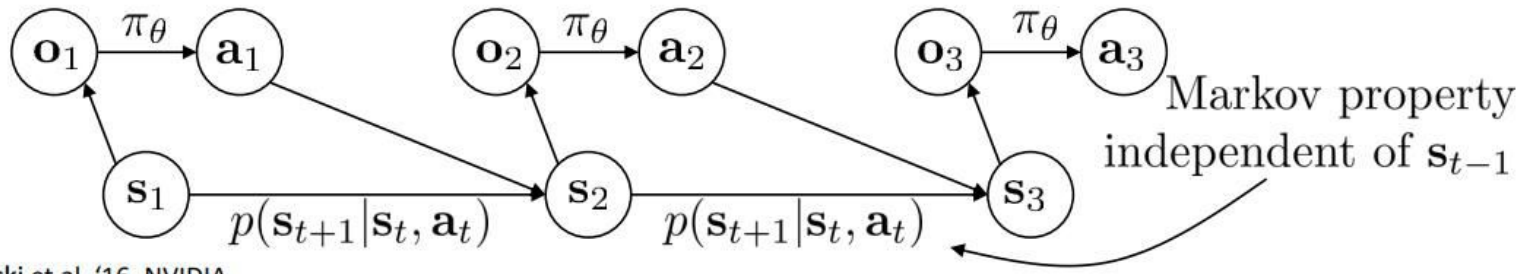
Example roll-out



$\mathbf{o}_t$ $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ $\mathbf{a}_t$

$\mathbf{s}_t$ – 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

Borrowed Slide, Here, observation provides another layer of uncertainty such that agent cannot fully grasp the current state.

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

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 briefly)
  - Estimate the optimal value function Q(s, a)
  - This is the maximum value achievable under any policy
- Model-based RL (will be discussed briefly)
  - Build a model of environment
  - Plan (e.g. by lookahead) using model
- SoA approaches generally combine flavors of all three.

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

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

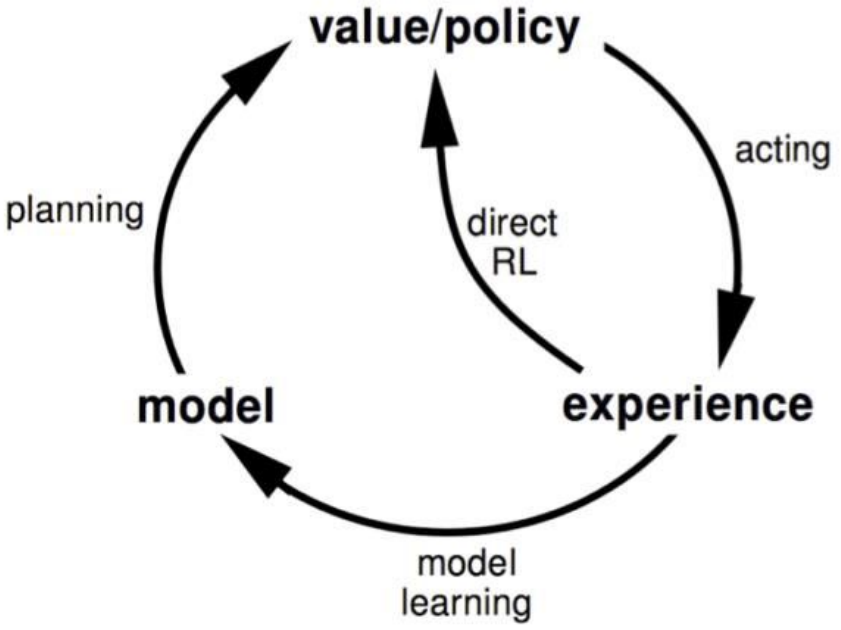
- Once we have optimal Q-value function we can act optimally

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

Model-based approach (in brief)

Intuition: If we know $\mathcal{F}$ and can evaluate $R(s)$ at any state, then we can just simulate the interaction with the environment and pick the action with the best outcome.

Build a mental image of $\mathcal{F}$ by learning



Policy-based approach

- Direct policy differentiation

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

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

Want to maximize this

How do we do it?

Policy-based approach

- Gradient Descent!

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

a convenient identity

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

Direct policy differentiation

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

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

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

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

log of both sides

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

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

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

Direct policy differentiation

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

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

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

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

log of both sides

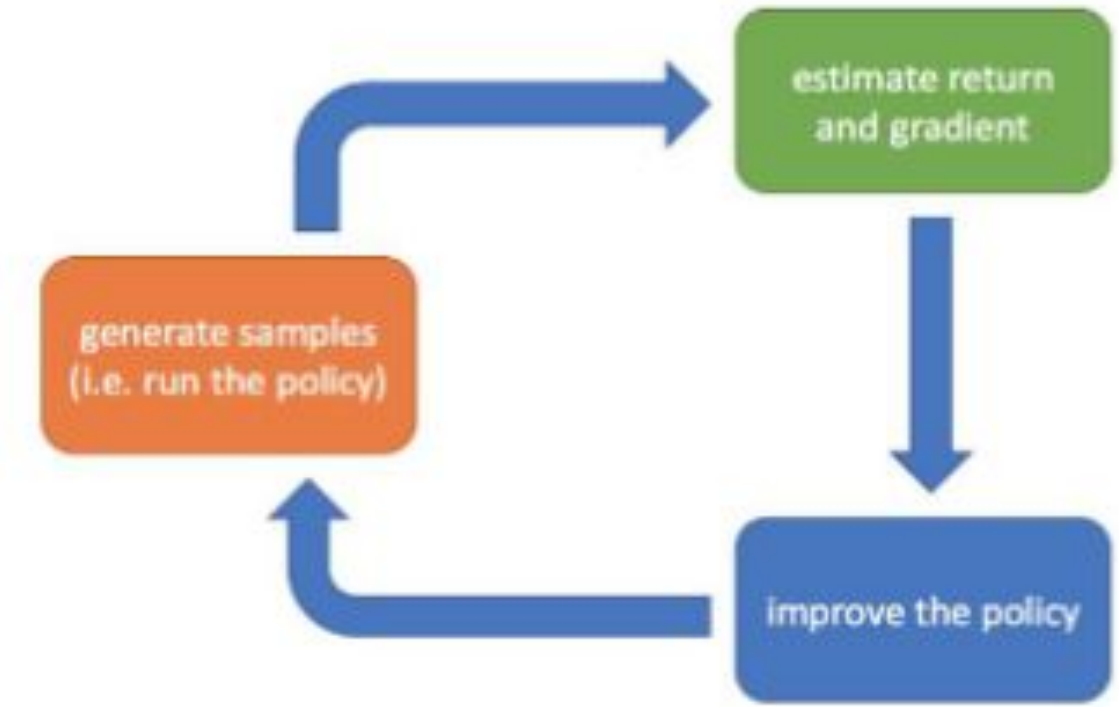
$$\log p_\theta(\tau) = \log p(\mathbf{s}_1) + \sum_{t=1}^{T}\log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

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

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

Computing the Policy Gradient

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



Evaluating the policy gradient

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

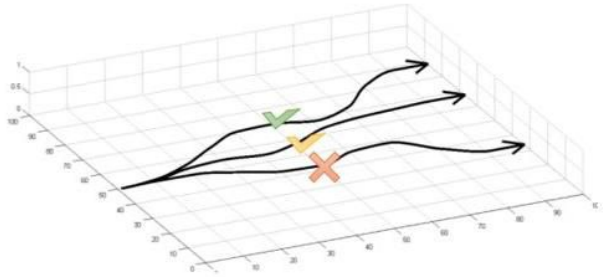
$$\nabla_\theta J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$

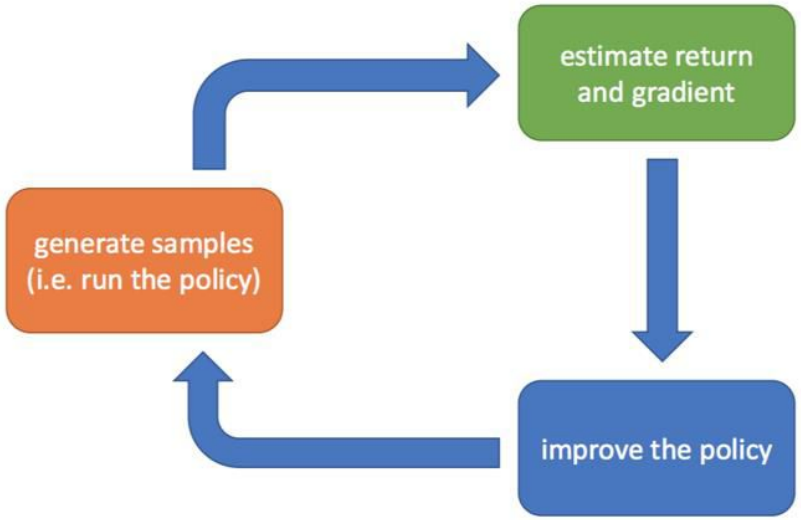
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

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

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





Further Reading

Policy Gradient Methods:

- Trust Region Policy Optimization https://arxiv.org/abs/1502.05477
- **Proximal Policy Optimization** https://arxiv.org/abs/1707.06347

Value Iteration Methods:

- Q learning, Temporal Difference, SARSA and etc. http://incompleteideas.net/book/bookdraft2017nov5.pdf
- Deep Q Networks https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf
- Deep Deterministic Policy Gradient (yes, this is more value learning than policy gradient) https://arxiv.org/abs/1509.02971
- **Soft Actor-Critic** https://arxiv.org/abs/1801.01290

Model Based:

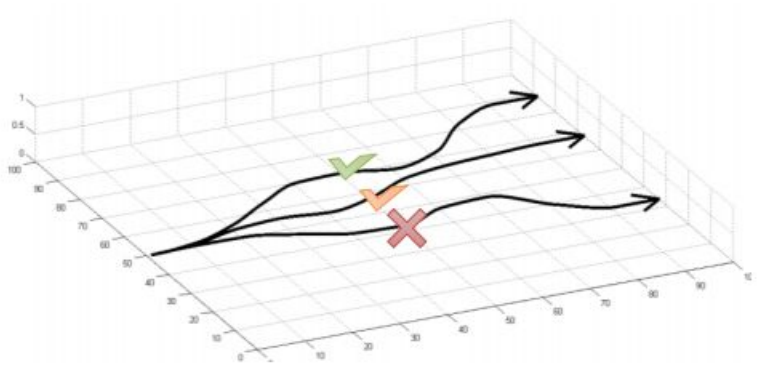
- Stochastic Lower-Bounds Optimization https://arxiv.org/abs/1807.03858
- (Nice review article that covers classical and newer methods) https://arxiv.org/abs/1907.02057
- AlphaGo https://deepmind.com/research/case-studies/alphago-the-story-so-far
- Alphazero https://arxiv.org/abs/1712.01815

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.

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

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 $t < t'$

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

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.