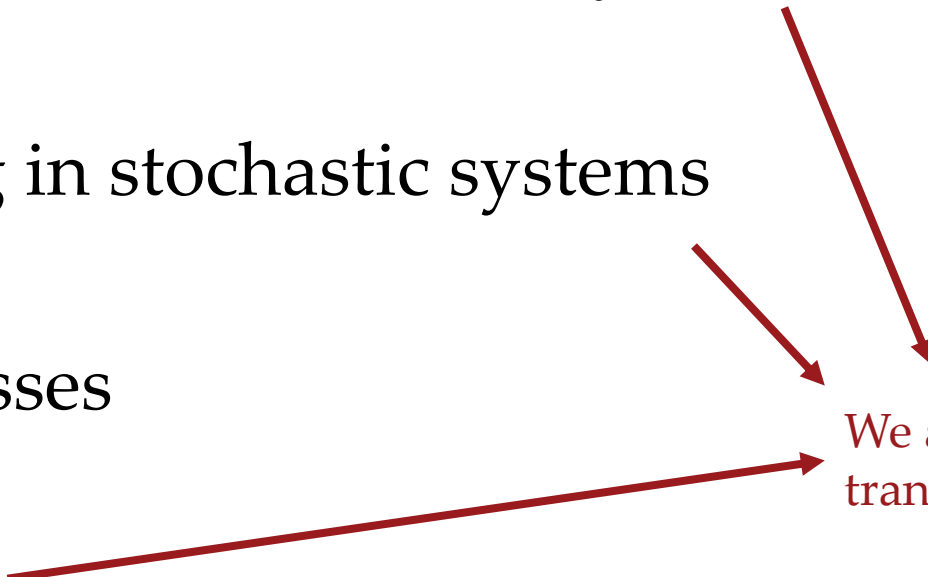


Robot Learning

Reinforcement learning

Last time...

- Dynamic programming in deterministic systems
- Dynamic programming in stochastic systems
- Markov decision processes
- Value/policy iteration



We assumed we know the transition function.

Today...

- Model-based reinforcement learning
- Model-free reinforcement learning

A great tutorial

ICML 2018 tutorial on “Optimization perspectives on learning to control” by Ben Recht:

https://youtu.be/hYw_qhLUE0o

Infinite horizon MDPs

State: $s \in \mathcal{S}$

Action: $a \in \mathcal{A}$

Transition: $s_{t+1} \sim P(\cdot | s_t, a_t)$

Reward: $r_t = R(s_t, a_t)$

Discount: $\gamma \in [0,1)$

Policy: $\pi: \mathcal{S} \rightarrow \mathcal{A}$ or $\pi: \mathcal{S} \rightarrow \Delta\mathcal{A}$

Goal:

$$\pi^* = \arg \max_{\pi} \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \right]$$

As a constrained optimization problem

$$\begin{aligned} & \underset{\pi}{\text{maximize}} && \mathbb{E}_{\mathbf{w}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right] \\ & \text{subject to} && s_t = f(s_t, a_t, w_t) \\ & && a_t = \pi(s_t) \end{aligned}$$

Now, what if we don't know the transition function f ?

As a constrained optimization problem

$$\begin{aligned} &\underset{\pi}{\text{maximize}} && \mathbb{E}_{\mathbf{w}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right] \\ &\text{subject to} && s_t = f(s_t, a_t, w_t) \\ & && a_t = \pi(s_t) \end{aligned}$$

✧ Reinforcement Learning ✧

Model-based

Model-free

Approximate DP
Direct Policy Search

Model-based RL

$$\begin{aligned} & \underset{\pi}{\text{maximize}} && \mathbb{E}_{\mathbf{w}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right] \\ & \text{subject to} && s_t = f(s_t, a_t, \mathbf{w}_t) \\ & && a_t = \pi(s_t) \end{aligned}$$

1. Collect some data from the environment: $(s_t, a_t, r_t, s_{t+1})_{t=1}^N$.
2. Use supervised learning to learn $\hat{f}$ and $\hat{R}$ (if not already known).
3. Solve the approximate problem assuming $\hat{f}$ and $\hat{R}$.

Approximate dynamic programming

$$\begin{aligned} &\underset{\pi}{\text{maximize}} && \mathbb{E}_{\mathbf{w}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right] \\ &\text{subject to} && s_t = f(s_t, a_t, w_t) \\ &&& a_t = \pi(s_t) \end{aligned}$$

Remember Bellman equation:

$$Q(s, a) = R(s, a) + \gamma \mathbb{E}_{s'|s,a} \left[\max_{a' \in \mathcal{A}} Q(s', a') \right]$$

Collect some data from environment and learn a Q -function.

Approximate dynamic programming

$$\text{maximize}_{\pi} \quad \mathbb{E}_{\mathbf{w}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right]$$

$$\begin{aligned} \text{subject to } s_t &= f(s_t, a_t, w_t) \\ a_t &= \pi(s_t) \end{aligned}$$

$$Q(s_t, a_t) \approx R(s_t, a_t) + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a')$$



$$Q_{\text{new}}(s_t, a_t) = (1 - \eta)Q_{\text{old}}(s_t, a_t) + \eta \left(R(s_t, a_t) + \gamma \max_{a' \in \mathcal{A}} Q_{\text{old}}(s_{t+1}, a') \right)$$

This is the SARSA algorithm.

Approximate dynamic programming

$$\text{maximize}_{\pi} \quad \mathbb{E}_{\mathbf{w}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right]$$

$$\text{subject to } s_t = f(s_t, a_t, w_t) \\ a_t = \pi(s_t)$$

$$Q(s_t, a_t) \approx R(s_t, a_t) + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a')$$



$$Q_{\text{new}}(s_t, a_t) = Q_{\text{old}}(s_t, a_t) + \eta \left(\underbrace{R(s_t, a_t) + \gamma \max_{a' \in \mathcal{A}} Q_{\text{old}}(s_{t+1}, a') - Q_{\text{old}}(s_t, a_t)}_{\text{TD error}} \right)$$

This is TD error. Many algorithms (e.g., DQN) use it.

Direct policy search

$$\begin{aligned} & \underset{\pi}{\text{maximize}} && \mathbb{E}_{\mathbf{w}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right] \\ & \text{subject to} && s_t = f(s_t, a_t, w_t) \\ & && a_t = \pi(s_t) \end{aligned}$$

Idea: Formulate it as an unconstrained optimization to solve for π .

But the set of possible π 's are too large. Instead, make it a stochastic policy with parameters θ .

Direct policy search

Objective to maximize: $J(\theta) = \mathbb{E}_{\tau \sim P_\theta(\tau)}[R(\tau)]$



$P_\theta(\tau)$ is the probability of trajectory τ under policy π_θ .



$R(\tau)$ is cumulative discounted return of trajectory τ .

Direct policy search

$$\begin{aligned} J(\theta) &= \mathbb{E}_{\tau \sim P_{\theta}(\tau)}[R(\tau)] \\ &= \int P_{\theta}(\tau) R(\tau) d\tau \end{aligned}$$

$$\begin{aligned} \nabla_{\theta} J(\theta) &= \int \nabla_{\theta} P_{\theta}(\tau) R(\tau) d\tau \\ &= \int R(\tau) \nabla_{\theta} P_{\theta}(\tau) d\tau \\ &= \int R(\tau) P_{\theta}(\tau) \frac{\nabla_{\theta} P_{\theta}(\tau)}{P_{\theta}(\tau)} d\tau \\ &= \int R(\tau) P_{\theta}(\tau) \nabla_{\theta} \log P_{\theta}(\tau) d\tau \\ &= \mathbb{E}_{\tau \sim P_{\theta}(\tau)}[R(\tau) \nabla_{\theta} \log P_{\theta}(\tau)] \end{aligned}$$

Direct policy search

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim P_{\theta}(\tau)} [R(\tau) \nabla_{\theta} \log P_{\theta}(\tau)]$$

$$\begin{aligned} \log P_{\theta}(\tau) &= \log \left(P(s_0) \prod_{t=0}^{\infty} P(s_{t+1} \mid s_t, a_t) \pi_{\theta}(a_t \mid s_t) \right) \\ &= \log P(s_0) + \sum_{t=0}^{\infty} \log P(s_{t+1} \mid s_t, a_t) + \sum_{t=0}^{\infty} \log \pi_{\theta}(a_t \mid s_t) \end{aligned}$$

$$\nabla_{\theta} \log P_{\theta}(\tau) = \sum_{t=0}^{\infty} \nabla_{\theta} \log \pi_{\theta}(a_t \mid s_t)$$

Direct policy search

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim P_{\theta}(\tau)} \left[\sum_{t=0}^{\infty} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right]$$

Because of causality:

More on this in
the homework!

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim P_{\theta}(\tau)} \left[\sum_{t=0}^{\infty} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \sum_{t'=t}^{\infty} \gamma^{t'} R(s_{t'}, a_{t'}) \right]$$

This is the REINFORCE algorithm. It is also known as policy gradient.

On-policy vs. off-policy

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim P_{\theta}(\tau)} \left[\sum_{t=0}^{\infty} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \sum_{t'=t}^{\infty} \gamma^{t'} R(s_{t'}, a_{t'}) \right]$$

 This is on-policy.

On-policy vs. off-policy

$$\begin{aligned} & \underset{\pi}{\text{maximize}} && \mathbb{E}_{\mathbf{w}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right] \\ & \text{subject to} && s_t = f(s_t, a_t, w_t) \\ & && a_t = \pi(s_t) \end{aligned}$$

1. Collect some data from the environment: $(s_t, a_t, r_t, s_{t+1})_{t=1}^N$.
2. Use supervised learning to learn $\hat{f}$ and $\hat{R}$ (if not already known).
3. Solve the approximate problem assuming $\hat{f}$ and $\hat{R}$.

On-policy vs. off-policy

$$\begin{aligned} &\underset{\pi}{\text{maximize}} && \mathbb{E}_{\mathbf{w}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right] \\ &\text{subject to} && s_t = f(s_t, a_t, w_t) \\ &&& a_t = \pi(s_t) \end{aligned}$$

$$Q(s_t, a_t) \approx R(s_t, a_t) + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a')$$



$$Q_{\text{new}}(s_t, a_t) = (1 - \eta)Q_{\text{old}}(s_t, a_t) + \eta \left(R(s_t, a_t) + \gamma \max_{a' \in \mathcal{A}} Q_{\text{old}}(s_{t+1}, a') \right)$$

This is the SARSA algorithm.

Today...

We relaxed the assumption that we have the transition model.

We still assume we have access to the reward function/samples.

Next time...

What if we do not have access to the reward function/samples but some expert trajectories?

- Imitation learning
- Inverse reinforcement learning