Reinforcement Learning

An Introduction

Brandon Kaplowitz

September 16, 2020

Outline

What is Reinforcement Learning (RL)?

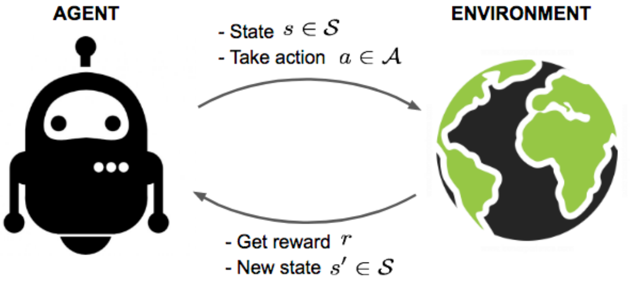
Reinforcement learning is learning what to do–how to map situations to actions–so as to maximize a numerical reward signal.

—Richard Sutton, Andrew Barto, Reinforcement Learning 2nd ed

What is Reinforcement Learning (RL)?

Reinforcement learning is learning what to do–how to map situations to actions–so as to maximize a numerical reward signal.

—*Richard Sutton, Andrew Barto, Reinforcement Learning 2nd ed*

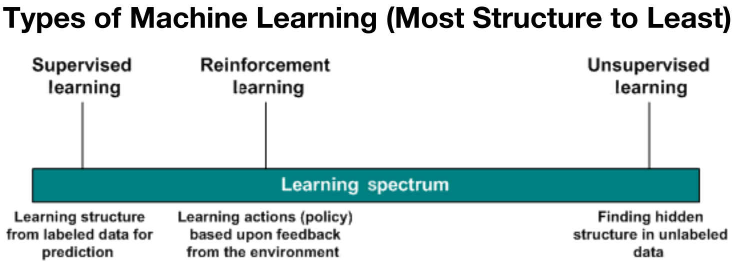


The RL algorithm loop. Source: https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html

`https://gym.openai.com/envs/CartPole-v1/`

Outline

How does Reinforcement Learning Relate to Machine Learning?



RL vs. Sup.: Difference is Ground truth not known. Courtesy: IBM

How does it compare to Bandits?

Main Difference: Environment

- **Reinforcement Learning**: Action affects future state

How does it compare to Bandits?

Main Difference: Environment

- **Reinforcement Learning**: Action affects future state
- **Bandits**: Action affects observables

How does it compare to Bandits?

Main Difference: Environment

- **Reinforcement Learning**: Action affects future state
- **Bandits**: Action affects observables
- **Online Learning**: Action affects reward

Outline

Definitions

- State Space $\mathcal{S}$, Action space $\mathcal{A}$, Observation Space $\mathcal{O}$, Reward Space $\mathcal{R} \subset \mathbb{R}$

Definitions

- State Space $\mathcal{S}$, Action space $\mathcal{A}$, Observation Space $\mathcal{O}$, Reward Space $\mathcal{R} \subset \mathbb{R}$
- **States**: $\mathbf{s}_t \in \mathcal{S}$ (book uses S_t, s', s for particular states)

Definitions

- State Space $\mathcal{S}$, Action space $\mathcal{A}$, Observation Space $\mathcal{O}$, Reward Space $\mathcal{R} \subset \mathbb{R}$
- **States**: $\mathbf{s}_t \in \mathcal{S}$ (book uses S_t, s', s for particular states)
- **Actions**: $\mathbf{a}_t \in \mathcal{A}$

Definitions

- State Space $\mathcal{S}$, Action space $\mathcal{A}$, Observation Space $\mathcal{O}$, Reward Space $\mathcal{R} \subset \mathbb{R}$
- **States**: $\mathbf{s}_t \in \mathcal{S}$ (book uses S_t, s', s for particular states)
- **Actions**: $\mathbf{a}_t \in \mathcal{A}$
- **Observables**: $\mathbf{o}_t \in \mathcal{O}$

Definitions

- State Space $\mathcal{S}$, Action space $\mathcal{A}$, Observation Space $\mathcal{O}$, Reward Space $\mathcal{R} \subset \mathbb{R}$
- **States**: $\mathbf{s}_t \in \mathcal{S}$ (book uses S_t, s', s for particular states)
- **Actions**: $\mathbf{a}_t \in \mathcal{A}$
- **Observables**: $\mathbf{o}_t \in \mathcal{O}$
- **Policy** function: $\pi_\theta(a_t|s_t)$ (θ indexing), $\pi_\theta : \mathcal{S} \to \mathcal{P}(\mathcal{A})$

Definitions

- State Space $\mathcal{S}$, Action space $\mathcal{A}$, Observation Space $\mathcal{O}$, Reward Space $\mathcal{R} \subset \mathbb{R}$
- **States**: $\mathbf{s}_t \in \mathcal{S}$ (book uses S_t, s', s for particular states)
- **Actions**: $\mathbf{a}_t \in \mathcal{A}$
- **Observables**: $\mathbf{o}_t \in \mathcal{O}$
- **Policy** function: $\pi_\theta(a_t|s_t)$ (θ indexing), $\pi_\theta : \mathcal{S} \to \mathcal{P}(\mathcal{A})$
- **Policy** function (if partially observed): $\pi(a_t|o_t)$.

Definitions

- State Space $\mathcal{S}$, Action space $\mathcal{A}$, Observation Space $\mathcal{O}$, Reward Space $\mathcal{R} \subset \mathbb{R}$
- **States**: $\mathbf{s}_t \in \mathcal{S}$ (book uses S_t, s', s for particular states)
- **Actions**: $\mathbf{a}_t \in \mathcal{A}$
- **Observables**: $\mathbf{o}_t \in \mathcal{O}$
- **Policy** function: $\pi_\theta(a_t|s_t)$ (θ indexing), $\pi_\theta : \mathcal{S} \to \mathcal{P}(\mathcal{A})$
- **Policy** function (if partially observed): $\pi(a_t|o_t)$.
- **Reward** (utility or loss) function: $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ (Could be $\to \mathcal{P}(\mathbb{R})$).

Definitions

- State Space $\mathcal{S}$, Action space $\mathcal{A}$, Observation Space $\mathcal{O}$, Reward Space $\mathcal{R} \subset \mathbb{R}$
- **States**: $\mathbf{s}_t \in \mathcal{S}$ (book uses S_t, s', s for particular states)
- **Actions**: $\mathbf{a}_t \in \mathcal{A}$
- **Observables**: $\mathbf{o}_t \in \mathcal{O}$
- **Policy** function: $\pi_\theta(a_t|s_t)$ (θ indexing), $\pi_\theta : \mathcal{S} \to \mathcal{P}(\mathcal{A})$
- **Policy** function (if partially observed): $\pi(a_t|o_t)$.
- **Reward** (utility or loss) function: $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ (Could be $\to \mathcal{P}(\mathbb{R})$).
- **Reward at time t** r_t (book uses R_t, r for particular realization of R_t)

Definitions

- State Space $\mathcal{S}$, Action space $\mathcal{A}$, Observation Space $\mathcal{O}$, Reward Space $\mathcal{R} \subset \mathbb{R}$
- **States**: $\mathbf{s}_t \in \mathcal{S}$ (book uses S_t, s', s for particular states)
- **Actions**: $\mathbf{a}_t \in \mathcal{A}$
- **Observables**: $\mathbf{o}_t \in \mathcal{O}$
- **Policy** function: $\pi_\theta(a_t|s_t)$ (θ indexing), $\pi_\theta : \mathcal{S} \to \mathcal{P}(\mathcal{A})$
- **Policy** function (if partially observed): $\pi(a_t|o_t)$.
- **Reward** (utility or loss) function: $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ (Could be $\to \mathcal{P}(\mathbb{R})$).
- **Reward at time t** r_t (book uses R_t, r for particular realization of R_t)
- **Return following time t** $G_t := \sum_{k=0}^{T} \gamma^{t+k+1} r_{t+k+1}$

Definitions

- State Space $\mathcal{S}$, Action space $\mathcal{A}$, Observation Space $\mathcal{O}$, Reward Space $\mathcal{R} \subset \mathbb{R}$
- **States**: $\mathbf{s}_t \in \mathcal{S}$ (book uses S_t, s', s for particular states)
- **Actions**: $\mathbf{a}_t \in \mathcal{A}$
- **Observables**: $\mathbf{o}_t \in \mathcal{O}$
- **Policy** function: $\pi_\theta(a_t|s_t)$ (θ indexing), $\pi_\theta : \mathcal{S} \to \mathcal{P}(\mathcal{A})$
- **Policy** function (if partially observed): $\pi(a_t|o_t)$.
- **Reward** (utility or loss) function: $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ (Could be $\to \mathcal{P}(\mathbb{R})$).
- **Reward at time t** r_t (book uses R_t, r for particular realization of R_t)
- **Return following time t** $G_t := \sum_{k=0}^{T} \gamma^{t+k+1} r_{t+k+1}$
- **Transition** operator $\mathcal{T}$ s.t. $\mathcal{T}_{i,j,k} = p(s_{t+1} = i | s_t = j, a_t = k)$, where

$$p(s_{t+1} = i) = \sum_{j,k} \mathcal{T}_{i,j,k} p(s_t = j) p(a_t = k)$$

Definitions

- State Space $\mathcal{S}$, Action space $\mathcal{A}$, Observation Space $\mathcal{O}$, Reward Space $\mathcal{R} \subset \mathbb{R}$
- **States**: $\mathbf{s}_t \in \mathcal{S}$ (book uses S_t, s', s for particular states)
- **Actions**: $\mathbf{a}_t \in \mathcal{A}$
- **Observables**: $\mathbf{o}_t \in \mathcal{O}$
- **Policy** function: $\pi_\theta(a_t|s_t)$ (θ indexing), $\pi_\theta : \mathcal{S} \to \mathcal{P}(\mathcal{A})$
- **Policy** function (if partially observed): $\pi(a_t|o_t)$.
- **Reward** (utility or loss) function: $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ (Could be $\to \mathcal{P}(\mathbb{R})$).
- **Reward at time t** r_t (book uses R_t, r for particular realization of R_t)
- **Return following time t** $G_t := \sum_{k=0}^{T} \gamma^{t+k+1} r_{t+k+1}$
- **Transition** operator $\mathcal{T}$ s.t. $\mathcal{T}_{i,j,k} = p(s_{t+1} = i | s_t = j, a_t = k)$, where

$$p(s_{t+1} = i) = \sum_{j,k} \mathcal{T}_{i,j,k} p(s_t = j) p(a_t = k)$$

- Markovian setting where $p(s_{t+1}|s^t) = p(s_{t+1}|s_t)$

Markov Decision Process

A Markov Decision Process (MDP) is

$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}$$

A Partially Observed Markov Decision Process (POMDP) is

$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{E}, r\},$$

where $\mathcal{E}$ is the *emission* probability:

$$\mathcal{E} = p(o_t | s_t)$$

.

Outline

Formal Problem

Note: marginal probability $p_\theta(\mathbf{s}_t, \mathbf{a}_t)$ indexed by θ since

$$
\begin{aligned}
p_\theta(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) &= \sum p(s_{t+1}, a_{t+1} | s_1, a_1, \ldots s_t, a_t) p(s_1, a_1, \ldots s_t, a_t) \\
&= \sum p(s_{t+1}, a_{t+1} | s_t, a_t) p(s_1, a_1, \ldots s_t, a_t) \\
&= \sum p(s_{t+1} | s_t, a_t) \pi_\theta(a_{t+1} | s_{t+1}) p(s_1, a_1, \ldots s_t, a_t) \\
&= \sum \pi_\theta(a_{t+1} | s_{t+1}) p(s_1) \prod_{j=1}^{t} \pi_\theta(a_t | s_t) p(s_{t+1} | s_t, a_t)
\end{aligned}
$$

Formal Problem

Note: marginal probability $p_\theta(\mathbf{s}_t, \mathbf{a}_t)$ indexed by θ since

$$
\begin{aligned}
p_\theta(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) &= \sum p(s_{t+1}, a_{t+1} | s_1, a_1, \ldots s_t, a_t) p(s_1, a_1, \ldots s_t, a_t) \\
&= \sum p(s_{t+1}, a_{t+1} | s_t, a_t) p(s_1, a_1, \ldots s_t, a_t) \\
&= \sum p(s_{t+1} | s_t, a_t) \pi_\theta(a_{t+1} | s_{t+1}) p(s_1, a_1, \ldots s_t, a_t) \\
&= \sum \pi_\theta(a_{t+1} | s_{t+1}) p(s_1) \prod_{j=1}^{t} \pi_\theta(a_t | s_t) p(s_{t+1} | s_t, a_t)
\end{aligned}
$$

Problem in full info case is in MDP environment find:

$$
\theta^* = \arg\max_\theta \mathbb{E}_{(s_t, a_t) \sim p_\theta(s_t, a_t)} \sum_{t=1}^{T} \gamma^t r(s_t, a_t)
$$

where $\gamma \in (0, 1)$ can be $= 1$ if $T < \infty$.

Formal Problem

Note: marginal probability $p_\theta(\mathbf{s}_t, \mathbf{a}_t)$ indexed by θ since

$$
\begin{aligned}
p_\theta(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) &= \sum p(s_{t+1}, a_{t+1} | s_1, a_1, \ldots s_t, a_t) p(s_1, a_1, \ldots s_t, a_t) \\
&= \sum p(s_{t+1}, a_{t+1} | s_t, a_t) p(s_1, a_1, \ldots s_t, a_t) \\
&= \sum p(s_{t+1} | s_t, a_t) \pi_\theta(a_{t+1} | s_{t+1}) p(s_1, a_1, \ldots s_t, a_t) \\
&= \sum \pi_\theta(a_{t+1} | s_{t+1}) p(s_1) \prod_{j=1}^{t} \pi_\theta(a_t | s_t) p(s_{t+1} | s_t, a_t)
\end{aligned}
$$

Problem in full info case is in MDP environment find:

$$
\theta^* = \arg\max_\theta \mathbb{E}_{(s_t, a_t) \sim p_\theta(s_t, a_t)} \sum_{t=1}^{T} \gamma^t r(s_t, a_t)
$$

where $\gamma \in (0, 1)$ can be $= 1$ if $T < \infty$. Equivalently to finding optimal policy func $\pi^* = \pi_{\theta^*}$.

Formal Problem

Note: marginal probability $p_\theta(\mathbf{s}_t, \mathbf{a}_t)$ indexed by θ since

$$
\begin{aligned}
p_\theta(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) &= \sum p(s_{t+1}, a_{t+1} | s_1, a_1, \ldots s_t, a_t) p(s_1, a_1, \ldots s_t, a_t) \\
&= \sum p(s_{t+1}, a_{t+1} | s_t, a_t) p(s_1, a_1, \ldots s_t, a_t) \\
&= \sum p(s_{t+1} | s_t, a_t) \pi_\theta(a_{t+1} | s_{t+1}) p(s_1, a_1, \ldots s_t, a_t) \\
&= \sum \pi_\theta(a_{t+1} | s_{t+1}) p(s_1) \prod_{j=1}^{t} \pi_\theta(a_t | s_t) p(s_{t+1} | s_t, a_t)
\end{aligned}
$$

Problem in full info case is in MDP environment find:

$$
\theta^* = \arg\max_\theta \mathbb{E}_{(s_t, a_t) \sim p_\theta(s_t, a_t)} \sum_{t=1}^{T} \gamma^t r(s_t, a_t)
$$

where $\gamma \in (0, 1)$ can be $= 1$ if $T < \infty$. Equivalently to finding optimal policy func $\pi^* = \pi_{\theta^*}$.

Definition: $\mathbb{E}_\pi$, expectation under $p_\theta(s_t, a_t)$ induced by $\pi = \pi_\theta$ for some θ.

Outline

Dynamic Programming

Classical **'complete knowledge'** MDP solution method (agent knows v and p_θ)

Usually impossible to solve analytically. We go for ε-*optimality*.

Dynamic Programming

Classical **'complete knowledge'** MDP solution method (agent knows v and p_θ)
Usually impossible to solve analytically. We go for ε-*optimality*.

State-Value function under policy π (assuming nonstochastic r):

$$v_\pi(s) := \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}, a_{t+k+1}) | s_t = s \right\} \quad \forall s \in \mathcal{S}$$

Dynamic Programming

Classical **'complete knowledge'** MDP solution method (agent knows v and p_θ)
Usually impossible to solve analytically. We go for ε-*optimality*.

State-Value function under policy π (assuming nonstochastic r):

$$v_\pi(s) := \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}, a_{t+k+1}) | s_t = s \right\} \quad \forall s \in \mathcal{S}$$

Bellman equation under policy π (assuming nonstochastic r) :

$$v_\pi(s) = \sum_a \pi(a|s) \left[r(s,a) + \sum_{s'} \gamma v_\pi(s') p(s'|s,a) \right] \quad \forall s \in \mathcal{S}$$

Dynamic Programming

Classical **'complete knowledge'** MDP solution method (agent knows v and p_θ)
Usually impossible to solve analytically. We go for ε-*optimality*.

State-Value function under policy π (assuming nonstochastic r):

$$v_\pi(s) := \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}, a_{t+k+1}) | s_t = s \right\} \quad \forall s \in \mathcal{S}$$

Bellman equation under policy π (assuming nonstochastic r) :

$$v_\pi(s) = \sum_a \pi(a|s) \left[r(s,a) + \sum_{s'} \gamma v_\pi(s') p(s'|s,a) \right] \quad \forall s \in \mathcal{S}$$

By *Bellman's optimality principle*, **optimal value function** under some optimal a:

$$v_*(s) = \max_a \left[r(s,a) + \sum_{s'} \gamma v_*(s') p(s'|s,a) \right] \quad \forall s \in \mathcal{S},$$

and optimal policy π_* by the $\arg\max$

Dynamic Programming

Classical **'complete knowledge'** MDP solution method (agent knows v and p_θ)
Usually impossible to solve analytically. We go for ε-*optimality*.

State-Value function under policy π (assuming nonstochastic r):

$$v_\pi(s) := \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}, a_{t+k+1}) | s_t = s \right\} \quad \forall s \in \mathcal{S}$$

Bellman equation under policy π (assuming nonstochastic r) :

$$v_\pi(s) = \sum_a \pi(a|s) \left[r(s,a) + \sum_{s'} \gamma v_\pi(s') p(s'|s,a) \right] \quad \forall s \in \mathcal{S}$$

By *Bellman's optimality principle*, **optimal value function** under some optimal a:

$$v_*(s) = \max_a \left[r(s,a) + \sum_{s'} \gamma v_*(s') p(s'|s,a) \right] \quad \forall s \in \mathcal{S},$$

and optimal policy π_* by the $\arg\max$ **Goal**: find $v_\pi \geq v_* - \varepsilon$ for given $\varepsilon > 0$.
Greedy one-step ahead approach given v_* for a.

Q function

Reminder:

$$v_\pi(s) := \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}, a_{t+k+1}) | s_t = s \right\} \quad \forall s \in \mathcal{S}$$

We also define:

$$q_\pi(s, a) := \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}, a_{t+k+1}) | s_t = s, a_t = a \right\} \quad \forall s \in \mathcal{S}, a \in \mathcal{A}$$

Q function

Reminder:

$$v_\pi(s) := \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}, a_{t+k+1}) | s_t = s \right\} \quad \forall s \in \mathcal{S}$$

We also define:

$$q_\pi(s,a) := \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}, a_{t+k+1}) | s_t = s, a_t = a \right\} \quad \forall s \in \mathcal{S}, a \in \mathcal{A}$$

action-value function: start from s, take action a, then follow π.

Q function

Reminder:

$$v_\pi(s) := \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}, a_{t+k+1}) | s_t = s \right\} \quad \forall s \in \mathcal{S}$$

We also define:

$$q_\pi(s,a) := \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}, a_{t+k+1}) | s_t = s, a_t = a \right\} \quad \forall s \in \mathcal{S}, a \in \mathcal{A}$$

action-value function: start from s, take action a, then follow π. Note that:

$$q_*(s,a) = r(s,a) + \sum_{s'} \gamma v_*(s') p(s'|s,a)$$

Q function

Reminder:

$$v_\pi(s) := \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}, a_{t+k+1}) | s_t = s \right\} \quad \forall s \in \mathcal{S}$$

We also define:

$$q_\pi(s,a) := \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}, a_{t+k+1}) | s_t = s, a_t = a \right\} \quad \forall s \in \mathcal{S}, a \in \mathcal{A}$$

action-value function: start from s, take action a, then follow π. Note that:

$$q_*(s,a) = r(s,a) + \sum_{s'} \gamma v_*(s') p(s'|s,a)$$

By Bellman's principle of optimality, we then get optimal Bellman equation:

$$q_*(s,a) = r(s,a) + \sum_{s'} \gamma \max_{a'} q_*(s',a') p(s'|s,a)$$

Q function

Reminder:

$$v_\pi(s) := \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}, a_{t+k+1}) | s_t = s \right\} \quad \forall s \in \mathcal{S}$$

We also define:

$$q_\pi(s,a) := \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}, a_{t+k+1}) | s_t = s, a_t = a \right\} \quad \forall s \in \mathcal{S}, a \in \mathcal{A}$$

action-value function: start from s, take action a, then follow π. Note that:

$$q_*(s,a) = r(s,a) + \sum_{s'} \gamma v_*(s') p(s'|s,a)$$

By Bellman's principle of optimality, we then get optimal Bellman equation:

$$q_*(s,a) = r(s,a) + \sum_{s'} \gamma \max_{a'} q_*(s',a') p(s'|s,a)$$

Don't need to know dynamics of model to solve for optimal a today!
Useful for future model-free RL lectures.

Policy Evaluation and Policy Improvement

- Prediction problem or policy evaluation takes a π and computes a v_π

Policy Evaluation and Policy Improvement

- Prediction problem or policy evaluation takes a π and computes a v_π
- One method is *Iterative policy evaluation*, computes v_π via the Bellman equation:

$$v_{k+1}(s) = \mathbb{E}_\pi \left[r + \gamma v_k(s_{t+1}) | s_t = s \right] \qquad v_0 \text{ guess given}, \quad \forall s \in S$$

Policy Evaluation and Policy Improvement

- Prediction problem or policy evaluation takes a π and computes a v_π
- One method is *Iterative policy evaluation*, computes v_π via the Bellman equation:

$$v_{k+1}(s) = \mathbb{E}_\pi \left[r + \gamma v_k(s_{t+1}) | s_t = s \right] \qquad v_0 \text{ guess given}, \;\; \forall s \in S$$

- Given v_π want π' s.t. $v_{\pi'}(s) \geq v_\pi(s) \ \forall s \in \mathcal{S}$

Policy Evaluation and Policy Improvement

- Prediction problem or policy evaluation takes a π and computes a v_π
- One method is *Iterative policy evaluation*, computes v_π via the Bellman equation:

$$v_{k+1}(s) = \mathbb{E}_\pi \left[r + \gamma v_k(s_{t+1}) | s_t = s \right] \qquad v_0 \text{ guess given}, \quad \forall s \in S$$

- Given v_π want π' s.t. $v_{\pi'}(s) \geq v_\pi(s)\ \forall s \in \mathcal{S}$
- *Policy improvement theorem* says we can do this by finding $\pi'(s)$ s.t.

$$q_\pi(s, \pi'(s)) \geq v_\pi(s)$$

or in other words we can find $\pi'(s)$ by

$$\pi'(s) = \arg\max_a \sum_{s'} p(s'|s,a) \left[r + \gamma v_\pi(s') \right]$$

Policy Evaluation and Policy Improvement

- Prediction problem or policy evaluation takes a π and computes a v_π
- One method is *Iterative policy evaluation*, computes v_π via the Bellman equation:

$$v_{k+1}(s) = \mathbb{E}_\pi \left[r + \gamma v_k(s_{t+1}) | s_t = s \right] \qquad v_0 \text{ guess given}, \quad \forall s \in S$$

- Given v_π want π' s.t. $v_{\pi'}(s) \geq v_\pi(s)\ \forall s \in \mathcal{S}$
- *Policy improvement theorem* says we can do this by finding $\pi'(s)$ s.t.

$$q_\pi(s, \pi'(s)) \geq v_\pi(s)$$

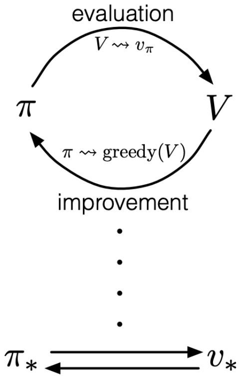
or in other words we can find $\pi'(s)$ by

$$\pi'(s) = \arg\max_a \sum_{s'} p(s'|s,a) \left[r + \gamma v_\pi(s') \right]$$

- Policy improvement: *Greedily* choose $\pi'(s)$ to maximize return today given v_π

Generalized Policy iteration

Generalized Policy Iteration: *Policy Improvement* interacting with *Prediction Problem*



Barto and Sutton, Introduction to Reinforcement Learning

Classical Solution Methods– Approximate Dynamic Programming

Wont go into these today as you all probably saw during first year:

- Value function iteration (start from value function and iterate)

Classical Solution Methods– Approximate Dynamic Programming

Wont go into these today as you all probably saw during first year:

- Value function iteration (start from value function and iterate)
- Policy function iteration (Howard Policy Improvement) (start from policy function and iterate)

Classical Solution Methods– Approximate Dynamic Programming

Wont go into these today as you all probably saw during first year:

- Value function iteration (start from value function and iterate)
- Policy function iteration (Howard Policy Improvement) (start from policy function and iterate)
- Linear Programming* (Optimizing twisted probability measures called "occupancy measures" of future state probabilities under different actions.)

Classical Solution Methods– Approximate Dynamic Programming

Wont go into these today as you all probably saw during first year:

- Value function iteration (start from value function and iterate)
- Policy function iteration (Howard Policy Improvement) (start from policy function and iterate)
- Linear Programming* (Optimizing twisted probability measures called "occupancy measures" of future state probabilities under different actions.)
- Planning/search based methods (future lecture?) Ex. shooting method and averaging over future simulations.

Outline

Tabular Reinforcement Learning

- We assumed r was determined *only* by the state and action pair.

Tabular Reinforcement Learning

- We assumed r was determined *only* by the state and action pair.
- This is called a tabular reinforcement learning environment

Tabular Reinforcement Learning

- We assumed r was determined *only* by the state and action pair.
- This is called a tabular reinforcement learning environment
- Given state s and action a can look up unique value V, reward r or action-value q. Basic setting for RL.

Tabular Reinforcement Learning

- We assumed r was determined *only* by the state and action pair.
- This is called a tabular reinforcement learning environment
- Given state s and action a can look up unique value V, reward r or action-value q. Basic setting for RL.

Tabular Reinforcement Learning

- We assumed r was determined *only* by the state and action pair.
- This is called a tabular reinforcement learning environment
- Given state s and action a can look up unique value V, reward r or action-value q. Basic setting for RL.

Initialized

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	327	0	0	0	0	0	0
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	499	0	0	0	0	0	0

Training

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	328	-2.30108105	-1.97092096	-2.30357004	-2.20591839	-10.3607344	-8.5583017
	. . .	. . .	. . .	. . .	. . .	. . .	. . .
	499	9.96984239	4.02706992	12.96022777	29	3.32877873	3.38230603

Courtesy: OpenAI Gym

Outline

Classical Model Based RL vs. Dynamic Programming

- Line between RL and classic DP methods from optimal control is fuzzy.

Classical Model Based RL vs. Dynamic Programming

- Line between RL and classic DP methods from optimal control is fuzzy.
- Primary distinction according to most is relaxation of assumption of complete knowledge of model dynamics/rewards and updating knowledge of model.

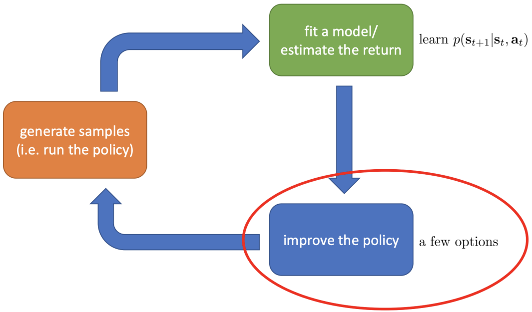
Classical Model Based RL vs. Dynamic Programming

- Line between RL and classic DP methods from optimal control is fuzzy.
- Primary distinction according to most is relaxation of assumption of complete knowledge of model dynamics/rewards and updating knowledge of model.
- $p_\theta(s, a)$ unknown and needs to be sampled from or fit.

Reinforcement Learning: A Few Algorithms

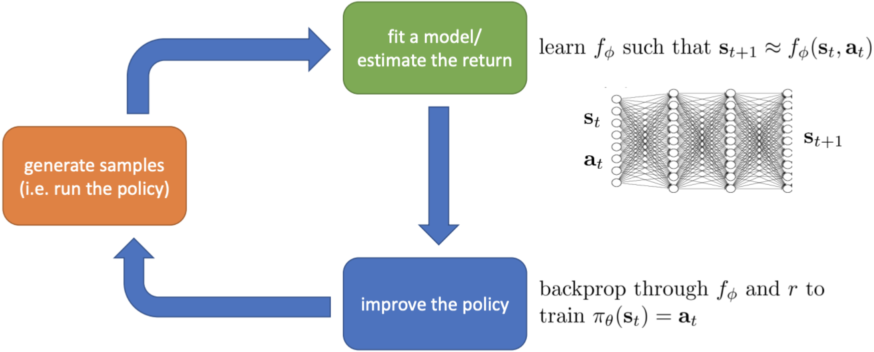
Organized More Structure to Less

Model-based RL algorithms



Model based RL. Courtesy: http://rail.eecs.berkeley.edu/deeprlcourse

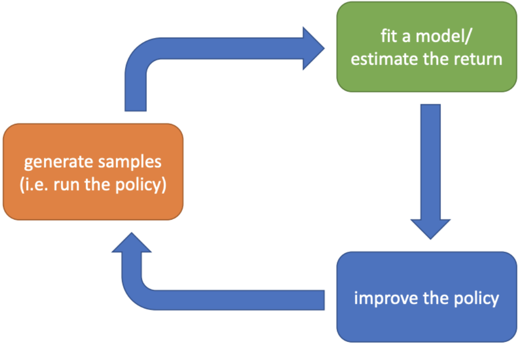
Another example: RL by backprop



Model based. Courtesy: http://rail.eecs.berkeley.edu/deeprlcourse

Reinforcement Learning: A Few Algorithms

Value function based algorithms



fit $V(\mathbf{s})$ or $Q(\mathbf{s}, \mathbf{a})$

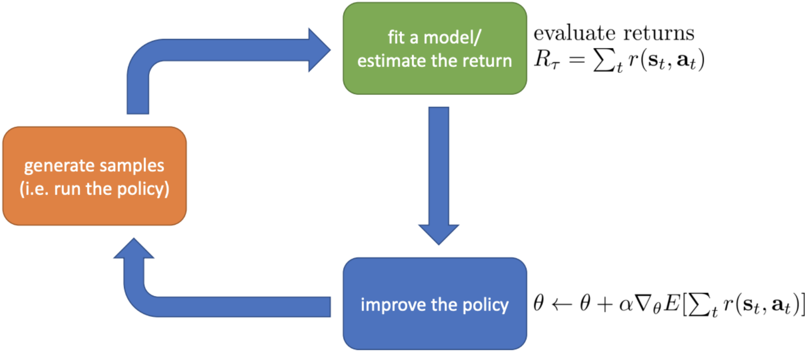
set $\pi(\mathbf{s}) = \arg\max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$

Model Free. Courtesy: http://rail.eecs.berkeley.edu/deeprlcourse

Outline

Reinforcement Learning: A Few Algorithms

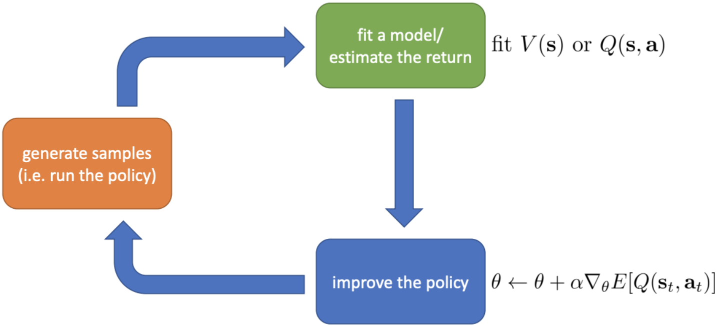
Direct policy gradients



Model Free. Courtesy: http://rail.eecs.berkeley.edu/deeprlcourse

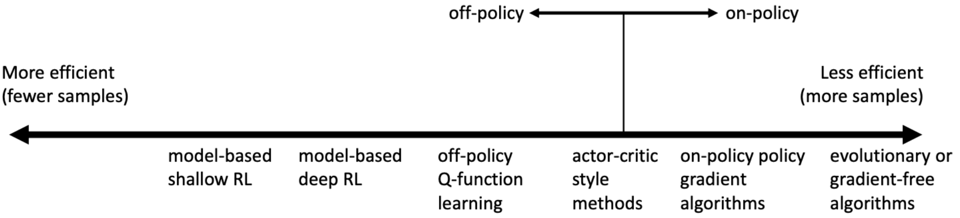
Reinforcement Learning: A Few Algorithms

Actor-critic: value functions + policy gradients



Actor Critic (Between Policy Gradient and Value Function). Courtesy: Berkeley, CS 285

Sample Efficiency and Structure



Courtesy: http://rail.eecs.berkeley.edu/deeprlcourse

Examples of specific algorithms

- Value function fitting methods
  - Q-learning, DQN
  - Temporal difference learning
  - Fitted value iteration

Will learn about some of these in future weeks.

Examples of specific algorithms

- Value function fitting methods
  - Q-learning, DQN
  - Temporal difference learning
  - Fitted value iteration
- Policy gradient methods
  - REINFORCE
  - Natural policy gradient
  - Trust region policy optimization

Will learn about some of these in future weeks.

Examples of specific algorithms

- Value function fitting methods
  - Q-learning, DQN
  - Temporal difference learning
  - Fitted value iteration
- Policy gradient methods
  - REINFORCE
  - Natural policy gradient
  - Trust region policy optimization
- Actor-critic algorithms
  - Asynchronous advantage actor-critic (A3C)
  - Soft actor-critic (SAC)

Will learn about some of these in future weeks.

Examples of specific algorithms

- Value function fitting methods
  - Q-learning, DQN
  - Temporal difference learning
  - Fitted value iteration
- Policy gradient methods
  - REINFORCE
  - Natural policy gradient
  - Trust region policy optimization
- Actor-critic algorithms
  - Asynchronous advantage actor-critic (A3C)
  - Soft actor-critic (SAC)
- Model-based RL algorithms
  - Dyna
  - Guided policy search

Will learn about some of these in future weeks.

Outline

General Setting

MDP Solution:

$$\theta^* = \arg\max_\theta \mathbb{E}_{(s_t,a_t)\sim p_\theta(s_t,a_t)} \mathbb{E}_{r_t \sim p_\theta(r_t|s_t,a_t)} \sum_{t=1}^{T} \gamma^t r_t$$

General Setting

MDP Solution:

$$\theta^* = \arg\max_\theta \mathbb{E}_{(s_t,a_t)\sim p_\theta(s_t,a_t)} \mathbb{E}_{r_t\sim p_\theta(r_t|s_t,a_t)} \sum_{t=1}^{T} \gamma^t r_t$$

Bellman Equation for State-Value (v) function:

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_*(s')\right]$$

General Setting

MDP Solution:

$$\theta^* = \arg\max_\theta \mathbb{E}_{(s_t,a_t)\sim p_\theta(s_t,a_t)} \mathbb{E}_{r_t\sim p_\theta(r_t|s_t,a_t)} \sum_{t=1}^{T} \gamma^t r_t$$

Bellman Equation for State-Value (v) function:

$$v_*(s) = \max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_*(s')\right]$$

Bellman Equation for Action-Value (q) function:

$$q_*(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma \max_{a'} q_*(s',a')\right]$$

Outline

Issues in Practice

- May not have perfect observability of underlying states.

Issues in Practice

- May not have perfect observability of underlying states.
  - Turn to algorithms for POMDPs.

Issues in Practice

- May not have perfect observability of underlying states.
  - Turn to algorithms for POMDPs.
- May need many samples especially for model free settings.

Issues in Practice

- May not have perfect observability of underlying states.
  - Turn to algorithms for POMDPs.
- May need many samples especially for model free settings.
  - Research into ‘sample optimal’ RL algorithms.

Issues in Practice

- May not have perfect observability of underlying states.
  - Turn to algorithms for POMDPs.
- May need many samples especially for model free settings.
  - Research into 'sample optimal' RL algorithms.
- Curse of dimmensionality. Increasing State + action space dimension leads to exponential increase in costs of exploring. Exponential number of states.

Issues in Practice

- May not have perfect observability of underlying states.
  - Turn to algorithms for POMDPs.
- May need many samples especially for model free settings.
  - Research into 'sample optimal' RL algorithms.
- Curse of dimmensionality. Increasing State + action space dimension leads to exponential increase in costs of exploring. Exponential number of states.
  - Tackled in Deep RL. Other feature selection/dimmensionality reduction methods

Issues in Practice

- May not have perfect observability of underlying states.
  - Turn to algorithms for POMDPs.
- May need many samples especially for model free settings.
  - Research into ‘sample optimal’ RL algorithms.
- Curse of dimmensionality. Increasing State + action space dimension leads to exponential increase in costs of exploring. Exponential number of states.
  - Tackled in Deep RL. Other feature selection/dimmensionality reduction methods
- Lack of sufficient data to train on.

Issues in Practice

- May not have perfect observability of underlying states.
  - Turn to algorithms for POMDPs.
- May need many samples especially for model free settings.
  - Research into ‘sample optimal’ RL algorithms.
- Curse of dimmensionality. Increasing State + action space dimension leads to exponential increase in costs of exploring. Exponential number of states.
  - Tackled in Deep RL. Other feature selection/dimmensionality reduction methods
- Lack of sufficient data to train on.
  - Train on self-play. (AlphaGo Zero). Offline reinforcement learning.

Outline

How Do We Design Intelligent _____ ?

- Machines?

How Do We Design Intelligent _____ ?

- Machines?
- AI?

How Do We Design Intelligent _____ ?

- Machines?
- AI?
- Model Agents?

How Do We Design Intelligent ______ ?

- Machines?
- AI?
- Model Agents?
- Inference?

How Do We Design Intelligent _____ ?

Key in all cases is that we are ‘*adaptive*’ to underlying changes in environment–exogenous or endogenously caused

Outline

Some examples of RL

Learning to Drive
(Courtesy: Wayve)

Some examples of RL

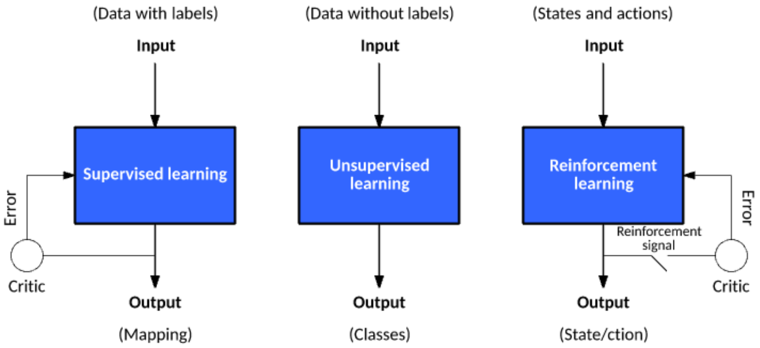
Hide and Seek
(Courtesy: OpenAI)

Some examples of RL

Alpha Go

Outline

How does Reinforcement Learning Relate?



Courtesy: IBM

What is Reinforcement Learning(RL) ?

Reinforcement learning is learning what to do-how to map situations to actions–so as to maximize a numerical reward signal. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them.

—Richard Sutton, Andrew Barto, Reinforcement Learning 2nd ed

What is Learning/Machine Learning?

Definition

Learning Algorithm (Mitchell 1997)
A computer program is said to *learn* from experience E with respect to a class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E

In ML:

What is Learning/Machine Learning?

Definition

Learning Algorithm (Mitchell 1997)
A computer program is said to *learn* from experience E with respect to a class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E

In ML:

- Task T is objective

What is Learning/Machine Learning?

Definition

Learning Algorithm (Mitchell 1997)

A computer program is said to *learn* from experience E with respect to a class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E

In ML:

- Task T is objective
- Performance P is measure of prediction ability (e.g. loss)

What is Learning/Machine Learning?

Definition

Learning Algorithm (Mitchell 1997)

A computer program is said to *learn* from experience E with respect to a class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E

In ML:

- Task T is objective
- Performance P is measure of prediction ability (e.g. loss)
- Experience E is some form of data (structured or not, labelled or not)

Machine Learning vs. Econometrics

- Fit and empirical performance vs. statistical properties or theoretical guarantees

Machine Learning vs. Econometrics

- Fit and empirical performance vs. statistical properties or theoretical guarantees
- Algorithms vs. estimation

Machine Learning vs. Econometrics

- Fit and empirical performance vs. statistical properties or theoretical guarantees
- Algorithms vs. estimation
- Not always clear cut... some work on theoretical guarantees in general environments. (Conformal prediction, algorithmic learning theory)

Machine Learning vs. Econometrics

- Fit and empirical performance vs. statistical properties or theoretical guarantees
- Algorithms vs. estimation
- Not always clear cut... some work on theoretical guarantees in general environments. (Conformal prediction, algorithmic learning theory)

Machine Learning vs. Econometrics

- Fit and empirical performance vs. statistical properties or theoretical guarantees
- Algorithms vs. estimation
- Not always clear cut... some work on theoretical guarantees in general environments. (Conformal prediction, algorithmic learning theory)

Chernozhukov: `https://arxiv.org/abs/1712.09089`
Athey: `https://arxiv.org/abs/1903.10075` (among many others)