Reinforcement Learning

1. Reinforcement learning problem: prediction and control

2. Some natural assumptions

3. Basic algorithm for control
Reinforcement Learning

1. Reinforcement learning problem: prediction and control

2. Some natural assumptions

3. Basic algorithm for control
Agent-Environment Interaction

Underlying MDP:

\[ \gamma = 0.9 \]

From current state, agent takes action. Environment (MDP) decides next state and reward. Possible history:

- \( s_2, \text{RED}, -2, s_3, \text{BLUE}, 1, s_1, \text{RED}, 0, s_1, \ldots \)

History conveys information about the MDP to the agent.
Agent-Environment Interaction

**Underlying MDP:**

```
γ = 0.9

s1  \(\rightarrow\) s2
  \[0.5, -1\]

s1  \(\rightarrow\) s3
  \[1, 2\]

s2  \(\rightarrow\) s3
  \[0.75, -2\]

s3  \(\rightarrow\) s1
  \[0.5, 3\]

0.5, 0  \(\rightarrow\) 1, 1
0.25, -1  \(\rightarrow\) 1, 1
0.5, 3  \(\rightarrow\) 0.75, -2
```

**Agent's view:**

```
γ = 0.9

s1

\(\rightarrow\)

s2

\(\rightarrow\)

s3
```

From current state, agent takes action. Environment (MDP) decides next state and reward. Possible history:

```
s2, RED, -2, s3, BLUE, 1, s1, RED, 0, s1, . . .
```

History conveys information about the MDP to the agent.
Agent-Environment Interaction

Underlying MDP:

- From current state, agent takes action.

Agent's view:

\( \gamma = 0.9 \)
Agent-Environment Interaction

Underlying MDP:

From current state, agent takes action.

Environment (MDP) decides next state and reward.
Agent-Environment Interaction

Underlying MDP:

\[
\begin{align*}
\text{Agent's view:} & \\
\gamma & = 0.9
\end{align*}
\]

- From current state, agent takes action.
- Environment (MDP) decides next state and reward.
- Possible history: \( s_2, \text{RED}, -2, s_3, \text{BLUE}, 1, s_1, \text{RED}, 0, s_1, \ldots \).
Agent-Environment Interaction

Underlying MDP:

From current state, agent takes action.

Environment (MDP) decides next state and reward.

Possible history: $s_2, \text{RED}, -2, s_3, \text{BLUE}, 1, s_1, \text{RED}, 0, s_1, \ldots$

History conveys information about the MDP to the agent.
The Control Problem

- For $t \geq 0$, let $h^t = (s^0, a^0, r^0, s^1, a^1, r^1, s^2, \ldots, s^t)$ denote a $t$-length history.
The Control Problem

- For $t \geq 0$, let $h^t = (s^0, a^0, r^0, s^1, a^1, r^1, s^2, \ldots, s^t)$ denote a $t$-length history.

- A learning algorithm $L$ is a mapping from the set of all histories to the set of all (probability distributions over) arms.
The Control Problem

- For $t \geq 0$, let $h^t = (s^0, a^0, r^0, s^1, a^1, r^1, s^2, \ldots, s^t)$ denote a $t$-length history.

- A learning algorithm $L$ is a mapping from the set of all histories to the set of all (probability distributions over) arms.

- Actions are selected by the learning algorithm (agent); next states and rewards are provided by the MDP (environment).
The Control Problem

- For $t \geq 0$, let $h^t = (s^0, a^0, r^0, s^1, a^1, r^1, s^2, \ldots, s^t)$ denote a $t$-length history.

- A learning algorithm $L$ is a mapping from the set of all histories to the set of all (probability distributions over) arms.

- Actions are selected by the learning algorithm (agent); next states and rewards are provided by the MDP (environment).

- **Control problem**: Can we construct $L$ such that

$$\lim_{H \to \infty} \frac{1}{H} \left( \sum_{t=0}^{H-1} \mathbb{P}\{a^t \sim L(h^t) \text{ is an optimal action for } s^t\} \right) = 1?$$
The Prediction Problem

- We are given a policy $\pi$ that the agent follows. The aim is to estimate $V^\pi$. 

For $t \geq 0$, let $h_t = (s_0, a_0, r_0, s_1, a_1, r_1, s_2, \ldots, s_t)$ denote a $t$-length history (note that $a_t \sim \pi(s_t)$).

A learning algorithm $L$ is a mapping from the set of all histories to the set of all mappings of the form $S \rightarrow \mathbb{R}$.

In other words, at each step $t$ the learning algorithm provides an estimate $\hat{V}_t$.

Prediction problem: Can we construct $L$ such that $\lim_{t \to \infty} \hat{V}_t = V^\pi$?
The Prediction Problem

- We are given a policy \( \pi \) that the agent follows. The aim is to estimate \( V^\pi \).

- For \( t \geq 0 \), let \( h^t = (s^0, a^0, r^0, s^1, a^1, r^1, s^2, \ldots, s^t) \) denote a \( t \)-length history (note that \( a^t \sim \pi(s^t) \)).
The Prediction Problem

- We are given a policy $\pi$ that the agent follows. The aim is to estimate $V^\pi$.

- For $t \geq 0$, let $h^t = (s^0, a^0, r^0, s^1, a^1, r^1, s^2, \ldots, s^t)$ denote a $t$-length history (note that $a^t \sim \pi(s^t)$).

- A learning algorithm $L$ is a mapping from the set of all histories to the set of all mappings of the form $S \rightarrow \mathbb{R}$.
The Prediction Problem

- We are given a policy \( \pi \) that the agent follows. The aim is to estimate \( V^\pi \).

- For \( t \geq 0 \), let \( h^t = (s^0, a^0, r^0, s^1, a^1, r^1, s^2, \ldots, s^t) \) denote a \( t \)-length history (note that \( a^t \sim \pi(s^t) \)).

- A learning algorithm \( L \) is a mapping from the set of all histories to the set of all mappings of the form \( S \rightarrow \mathbb{R} \).

- In other words, at each step \( t \) the learning algorithm provides an estimate \( \hat{V}^t \).
The Prediction Problem

- We are given a policy $\pi$ that the agent follows.
- The aim is to estimate $V^\pi$.

For $t \geq 0$, let $h^t = (s^0, a^0, r^0, s^1, a^1, r^1, s^2, \ldots, s^t)$ denote a $t$-length history (note that $a^t \sim \pi(s^t)$).

A learning algorithm $L$ is a mapping from the set of all histories to the set of all mappings of the form $S \rightarrow \mathbb{R}$.

In other words, at each step $t$ the learning algorithm provides an estimate $\hat{V}^t$.

Prediction problem: Can we construct $L$ such that

$$\lim_{t \rightarrow \infty} \hat{V}^t = V^\pi?$$
Reinforcement Learning

1. Reinforcement learning problem: prediction and control

2. Some natural assumptions

3. Basic algorithm for control
Assumption 1: Irreducibility

- Fix an MDP $M = (S, A, T, R, \gamma)$ and a policy $\pi$.
- Draw a graph with states as vertices and every non-zero-probability transition under $\pi$ as a directed edge.
- Is there a directed path from $s$ to $s'$ for every $s, s' \in S$?
- If yes, $M$ is irreducible under $\pi$.
- If $M$ is irreducible under all $\pi \in \Pi$, then $M$ is irreducible.
Assumption 2: Aperiodicity

- Fix an MDP $M = (S, A, T, R, \gamma)$ and a policy $\pi$.
- For $s \in S$, $t \geq 1$, let $X(s, t)$ be the set of all states $s'$ s. t. there is a non-zero probability of reaching $s'$ in exactly $t$ steps by starting at $s$ and following $\pi$.
- For $s \in S$, let $Y(s)$ be the set of all $t \geq 1$ such that $s \in X(s, t)$; let $p(s) = \gcd(Y(s))$.
- $M$ is aperiodic under $\pi$ if for all $s \in S$: $p(s) = 1$.
- If $M$ is aperiodic under all $\pi \in \Pi$, then $M$ is aperiodic.

$Y(s_1) = \{2, 4, 6, \ldots \}$.
Periodic.

$Y(s_1) = \{1, 2, 3, \ldots \}$.
$Y(s_2) = \{2, 3, 4, \ldots \}$.
Aperiodic.
Ergodicity

- An MDP that is irreducible and aperiodic is called an \textit{ergodic} MDP.
Ergodicity

- An MDP that is irreducible and aperiodic is called an **ergodic** MDP.

- In an ergodic MDP, every policy $\pi$ induces a unique steady state distribution $\mu^\pi : S \to (0, 1)$, subject to $\sum_{s \in S} \mu^\pi(s) = 1$, which is independent of the start state.
Ergodicity

- An MDP that is irreducible and aperiodic is called an **ergodic** MDP.

- In an ergodic MDP, every policy $\pi$ induces a unique steady state distribution $\mu^\pi : S \rightarrow (0, 1)$, subject to $\sum_{s \in S} \mu^\pi(s) = 1$, which is independent of the start state.

- For $s \in S$, $t \geq 0$, let $p(s, t)$ be the probability of being in state $s$ at step $t$, after starting at some (arbitrarily) fixed state and following $\pi$. 
Ergodicity

- An MDP that is irreducible and aperiodic is called an ergodic MDP.

- In an ergodic MDP, every policy \( \pi \) induces a unique steady state distribution \( \mu^\pi : S \to (0, 1) \), subject to \( \sum_{s \in S} \mu^\pi(s) = 1 \), which is independent of the start state.

- For \( s \in S \), \( t \geq 0 \), let \( p(s, t) \) be the probability of being in state \( s \) at step \( t \), after starting at some (arbitrarily) fixed state and following \( \pi \). Then

\[
\mu^\pi(s) = \lim_{t \to \infty} p(s, t).
\]
Ergodicity

- An MDP that is irreducible and aperiodic is called an **ergodic** MDP.

- In an ergodic MDP, every policy $\pi$ induces a unique steady state distribution $\mu^\pi : S \to (0, 1)$, subject to $\sum_{s \in S} \mu^\pi(s) = 1$, which is independent of the start state.

- For $s \in S$, $t \geq 0$, let $p(s, t)$ be the probability of being in state $s$ at step $t$, after starting at some (arbitrarily) fixed state and following $\pi$. Then

$$
\mu^\pi(s) = \lim_{t \to \infty} p(s, t).
$$

- We’ll use ergodicity in some of the later lectures.
1. Reinforcement learning problem: prediction and control

2. Some natural assumptions

3. Basic algorithm for control
A Model-based Approach

- A **model** is an estimate of the MDP, which is usually updated based on experience. We keep estimates $\hat{T}$ and $\hat{R}$, and try to get them to converge to $T$ and $R$, respectively.
A Model-based Approach

- A **model** is an estimate of the MDP, which is usually updated based on experience. We keep estimates $\hat{T}$ and $\hat{R}$, and try to get them to converge to $T$ and $R$, respectively.

- At convergence, acting optimally for MDP $(S, A, \hat{T}, \hat{R}, \gamma)$ must be optimal for the original MDP $(S, A, T, R, \gamma)$, too.
A Model-based Approach

- A model is an estimate of the MDP, which is usually updated based on experience. We keep estimates $\hat{T}$ and $\hat{R}$, and try to get them to converge to $T$ and $R$, respectively.

- At convergence, acting optimally for MDP $(S, A, \hat{T}, \hat{R}, \gamma)$ must be optimal for the original MDP $(S, A, T, R, \gamma)$, too.

- We must visit every state-action pair infinitely often.
A Model-based Approach

- A model is an estimate of the MDP, which is usually updated based on experience. We keep estimates $\hat{T}$ and $\hat{R}$, and try to get them to converge to $T$ and $R$, respectively.

- At convergence, acting optimally for MDP $(S, A, \hat{T}, \hat{R}, \gamma)$ must be optimal for the original MDP $(S, A, T, R, \gamma)$, too.

- We must visit every state-action pair infinitely often.

- Remember GLIE?
Algorithm

**Model-based RL**

//Initialisation
For \( s, s' \in S, a \in A \):
\[
\hat{T}[s][a][s'] \leftarrow 0; \hat{R}[s][a][s'] \leftarrow 0.
\]
For \( s, s' \in S, a \in A \):
\[
\text{totalTransitions}[s][a][s'] \leftarrow 0;
\text{totalReward}[s][a][s'] \leftarrow 0.
\]
For \( s \in S, a \in A \):
\[
\text{totalVisits}[s][a] \leftarrow 0.
\]
\( \text{modelValid} \leftarrow \text{False}. \)

Assume that the agent is born in state \( s^0 \). //Continued on next slide.
Assume that the agent is born in state $s^0$. //Continued from previous slide.

//For ever
For $t = 0, 1, 2, \ldots$:
    If $\text{modelValid}$:
        $\pi^{\text{opt}} \leftarrow \text{MDPPlan}(S, A, \hat{T}, \hat{R}, \gamma)$.
        $a^t \leftarrow \begin{cases} 
        \pi^{\text{opt}}(s^t) & \text{w. p. } 1 - \epsilon_t, \\
        \text{UniformRandom}(A) & \text{w. p. } \epsilon_t.
        \end{cases}$
    Else:
        $a^t \leftarrow \text{UniformRandom}(A)$.

Take action $a^t$; obtain reward $r^t$, next state $s^{t+1}$.
$\text{UpdateModel}(s^t, a^t, r^t, s^{t+1})$. 
**Algorithm**

```plaintext
UpdateModel(s, a, r, s')

\[ totalTransitions[s][a][s'] \leftarrow totalTransitions[s][a][s'] + 1. \]
\[ totalReward[s][a][s'] \leftarrow totalReward[s][a][s'] + r. \]
\[ totalVisits[s][a] \leftarrow totalVisits[s][a] + 1. \]

For \( s'' \in S \):
\[ \hat{T}[s][a][s''] \leftarrow \frac{totalTransitions[s][a][s'']}{totalVisits[s][a]}. \]
\[ \hat{R}[s][a][s'] \leftarrow \frac{totalReward[s][a][s']}{totalTransitions[s][a][s']}. \]

If \( \neg \text{modelValid} \):
  If \( \forall s'' \in S, \forall a'' \in A : totalVisits[s''][a''] \geq 1 \):
    \[ \text{modelValid} \leftarrow \text{True}. \]
```
Discussion

- Algorithm takes a sub-linear number of sub-optimal actions. Can still be optimised in many ways (computational complexity, exploration, etc.).
Discussion

- Algorithm takes a sub-linear number of sub-optimal actions. Can still be optimised in many ways (computational complexity, exploration, etc.).

- For convergence to optimal behaviour, does the algorithm need irreducibility and aperiodicity?
Discussion

- Algorithm takes a sub-linear number of sub-optimal actions. Can still be optimised in many ways (computational complexity, exploration, etc.).

- For convergence to optimal behaviour, does the algorithm need irreducibility and aperiodicity?

- Why is this a “model-based” algorithm?
Discussion

- Algorithm takes a sub-linear number of sub-optimal actions. Can still be optimised in many ways (computational complexity, exploration, etc.).

- For convergence to optimal behaviour, does the algorithm need irreducibility and aperiodicity?

- Why is this a “model-based” algorithm?
  Uses $\Theta(|S|^2|A|)$ memory. Will soon see a “model-free” method that needs $\Theta(|S||A|)$ memory.
1. Reinforcement Learning problem: prediction and control

2. Some natural assumptions

3. Basic algorithm for control
Reinforcement Learning

1. Reinforcement Learning problem: prediction and control

2. Some natural assumptions

3. Basic algorithm for control

Next week: some approaches for prediction.