Reinforcement learning

Basics:

- **Learner interacts with the environment**
  - Receives input with information about the environment (e.g. from sensors)
  - Makes actions that (may) effect the environment
  - Receives a reinforcement signal that provides a feedback on how well it performed
Reinforcement learning

**Objective:** Learn how to act in the environment in order to maximize the reinforcement signal

- The selection of actions should depend on the input
- A policy \( \pi : X \rightarrow A \) maps inputs to actions
- **Goal:** find the optimal policy \( \pi : X \rightarrow A \) that gives the best expected reinforcements

Example: learn how to play games (AlphaGo)

Gambling example

- **Game:** 3 biased coins
  - The coin to be tossed is selected randomly from the three coin options. The agent always sees which coin is going to be played next. The agent makes a bet on either a head or a tail with a wage of $1. If after the coin toss, the outcome agrees with the bet, the agent wins $1, otherwise it loses $1
- **RL model:**
  - **Input:** X – a coin chosen for the next toss,
  - **Action:** A – choice of head or tail the agent bets on,
  - **Reinforcements:** \{1, -1\}
- **A policy** \( \pi : X \rightarrow A \)
- Example:
  \[
  \begin{array}{c|c|c}
  \text{Coin} & \rightarrow & \text{head} \\
  1 & \rightarrow & \text{tail} \\
  2 & \rightarrow & \text{head} \\
  \end{array}
  \]
Gambling example

**RL model:**
- **Input:** $X$ – a coin chosen for the next toss,
- **Action:** $A$ – choice of head or tail the agent bets on,
- **Reinforcements:** $\{1, -1\}$
- **A policy** $\pi$: $\begin{pmatrix}
  \text{Coin1} & \rightarrow & \text{head} \\
  \text{Coin2} & \rightarrow & \text{tail} \\
  \text{Coin3} & \rightarrow & \text{head}
\end{pmatrix}$

**State, action reward trajectories**

<table>
<thead>
<tr>
<th>Step 0</th>
<th>Step 1</th>
<th>Step 2</th>
<th>Step k</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>Coin2</td>
<td>Coin1</td>
<td>Coin2</td>
</tr>
<tr>
<td>action</td>
<td>Tail</td>
<td>Head</td>
<td>Tail</td>
</tr>
<tr>
<td>reward</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Gambling example

**Learning goal:** find the optimal policy

$\pi^*: X \rightarrow A$

$\pi^*$: $\begin{pmatrix}
  \text{Coin1} & \rightarrow & ? \\
  \text{Coin2} & \rightarrow & ? \\
  \text{Coin3} & \rightarrow & ?
\end{pmatrix}$

$\pi^*:$ $\begin{pmatrix}
  \text{Coin1} & \rightarrow & ? \\
  \text{Coin2} & \rightarrow & ? \\
  \text{Coin3} & \rightarrow & ?
\end{pmatrix}$

maximizing future expected rewards

$$E(\sum_{t=0}^{T} \gamma^t r_t) \quad 0 \leq \gamma < 1$$

a discount factor = present value of money
**Expected rewards**

- Expected rewards for $\pi : X \rightarrow A$

  $E(\sum_{t=0}^{T} r_t)$  
  Expectation over many possible reward trajectories defined by $\pi : X \rightarrow A$

**Expected discounted rewards**

- Expected discounting rewards for $\pi : X \rightarrow A$
- Discounting with $0 \leq \gamma < 1$ (future value of money)

**No discounting:**

- $E(\sum_{t=0}^{T} r_t)$  
  Expectation over many possible discounted reward trajectories for $\pi : X \rightarrow A$
RL learning: objective functions

- **Objective:**
  Find a policy \( \pi^* : X \rightarrow A \)
  That maximizes some combination of future reinforcements (rewards) received over time

- **Valuation models** (quantify how good the mapping is):
  - **Finite horizon models**
    \[
    E\left(\sum_{t=0}^{T} r_t \right) \quad \text{Time horizon: } T > 0
    \]
    \[
    E\left(\sum_{t=0}^{T} \gamma^t r_t \right) \quad \text{Discount factor: } 0 \leq \gamma < 1
    \]
  - **Infinite horizon discounted model**
    \[
    E\left(\sum_{t=0}^{\infty} \gamma^t r_t \right) \quad \text{Discount factor: } 0 \leq \gamma < 1
    \]
  - **Average reward**
    \[
    \lim_{T \rightarrow \infty} \frac{1}{T} E\left(\sum_{t=0}^{T} r_t \right)
    \]

Agent navigation example

- **Agent navigation in the maze:**
  - 4 moves in compass directions
  - Effects of moves are stochastic – we may wind up in other than intended location with a non-zero probability
  - **Objective:** learn how to reach the goal state in the shortest expected time

Agent navigation example

- The RL model:
  - **Input**: $X$ – a position of an agent
  - **Output**: $A$ – the next move
  - **Reinforcements**: $R$
    - -1 for each move
    - +100 for reaching the goal
  - A policy: $\pi : X \rightarrow A$
    - Position 1 $\rightarrow$ right
    - Position 2 $\rightarrow$ right
    - ... (omitted)
    - Position 25 $\rightarrow$ left

- Goal: find the policy maximizing future expected rewards
  $$E(\sum_{t=0}^{\infty} \gamma^t r_t) \quad 0 \leq \gamma < 1$$

Agent navigation example

State, action reward trajectories

- policy
  $$\pi : \begin{array}{l}
  \text{Position 1} \rightarrow \text{right} \\
  \text{Position 2} \rightarrow \text{right} \\
  \ldots \\
  \text{Position 25} \rightarrow \text{left}
  \end{array}$$

<table>
<thead>
<tr>
<th>state</th>
<th>action</th>
<th>reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pos1</td>
<td>Right</td>
<td>-1</td>
</tr>
<tr>
<td>Pos2</td>
<td>Right</td>
<td>-1</td>
</tr>
<tr>
<td>Pos3</td>
<td>Up</td>
<td>-1</td>
</tr>
<tr>
<td>Pos15</td>
<td>Up</td>
<td>-1</td>
</tr>
<tr>
<td>Pos25</td>
<td>Left</td>
<td>-1</td>
</tr>
</tbody>
</table>

Step 0: Pos1 Right -1
Step 1: Pos2 Right -1
Step 2: Pos3 Up -1
Step k: Pos15 Up -1...
Effects of actions on the environment

Effect of actions on the environment
– More specifically on the next input \( x \) to be seen

**Case 1. No effect.** The distribution over possible \( x \) is independent of past actions. The rewards received depend only on the current state \( x \) and the action \( a \) chosen.

- **Reinforcement learning with immediate rewards**
  – 3 coin example
    What coin we see next is not affected by our previous action, hence our action does not effect future rewards

<table>
<thead>
<tr>
<th>Step</th>
<th>Coin 2</th>
<th>Coin 1</th>
<th>Coin 2</th>
<th>Coin 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>Tail</td>
<td>Head</td>
<td>Tail</td>
<td>Head</td>
</tr>
<tr>
<td>Action</td>
<td>Tail</td>
<td>Head</td>
<td>Tail</td>
<td>Head</td>
</tr>
<tr>
<td>Reward</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Effects of actions on the environment

Effect of actions on the environment
– More specifically on the next input \( x \) to be seen

**Case 2. Actions may effect the environment** and next inputs \( x \). The distribution of \( x \) can change due to past actions; the rewards related to the action can be seen with some delay.

- **Learning with delayed rewards**
  – Agent navigation example; a move action effects next position, and hence more distant future rewards

<table>
<thead>
<tr>
<th>Step</th>
<th>Pos 1</th>
<th>Pos 2</th>
<th>Pos 3</th>
<th>Pos 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>Right</td>
<td>Right</td>
<td>Up</td>
<td>Up</td>
</tr>
<tr>
<td>Action</td>
<td>Right</td>
<td>Right</td>
<td>Up</td>
<td>Up</td>
</tr>
<tr>
<td>Reward</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>
**RL with immediate rewards**

- **Game:** 3 biased coins
  - The coin to be tossed is selected randomly from the three coin options. The agent always sees which coin is going to be played next. The agent makes a bet on either a head or a tail with a wage of $1. If after the coin toss, the outcome agrees with the bet, the agent wins $1, otherwise it loses $1

- **RL model:**
  - Input: \( X \) – a coin chosen for the next toss
  - Action: \( A \) – head or tail the agent bets on
  - Reinforcements: \( \{1, -1\} \) ($1 either won or lost)

- **Learning goal:** find the optimal policy \( \pi^* : X \rightarrow A \) maximizing the future expected profits over time

\[
E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right) \quad 0 \leq \gamma < 1 \quad \text{a discount factor}
\]

---

**RL with immediate rewards**

- **Expected reward**
\[
E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right) \quad 0 \leq \gamma < 1
\]

- **Immediate reward case:**
  - Reward depends only on \( x \) and the action choice
  - The action does not affect the environment and hence future inputs (states) and future rewards:

<table>
<thead>
<tr>
<th>General Trajectory</th>
<th>Step0</th>
<th>Step1</th>
<th>Step2</th>
<th>Step k</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>( x_0 )</td>
<td>( x_1 )</td>
<td>( x_2 )</td>
<td>( x_k )</td>
</tr>
<tr>
<td>action</td>
<td>( a_0 )</td>
<td>( a_1 )</td>
<td>( a_2 )</td>
<td>..</td>
</tr>
<tr>
<td>Reward</td>
<td>( r_{x0,a0} )</td>
<td>( r_{x1,a1} )</td>
<td>( r_{x2,a2} )</td>
<td>( r_{xk,ak} )</td>
</tr>
</tbody>
</table>

\[
E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right) = E(r_{x0,a0}) + \gamma E(r_{x1,a1}) + \gamma^2 E(r_{x2,a2}) + ... E(\gamma^k r_{xk,ak}) + ... \\
_{
= E(r_{x0,a0}) + \gamma E(r_{x1,a1}) + \gamma^2 E(r_{x2,a2}) + ... \gamma^k E(r_{xk,ak}) + ...
\]
**RL with immediate rewards**

**Immediate reward case:**
- Reward for input $x$ and the action choice $a$ may vary
- **Expected one-step reward for the input $x$ and action $a$:**
  \[ R(x, a) = E(r_{x,a}) \]
  - For the coin bet problem it is:
  \[ R(x, a_i) = \sum r(\omega_j | a_i, x) P(\omega_j | x, a_i) \]
  $\omega_j$ : an outcome of the coin toss $x$
  $r(\omega_j | a_i, x)$ : reward for an outcome and the bet made on $x$
- **Expected one step reward for a policy** $\pi : X \rightarrow A$
  \[ R(x, \pi(x)) = E(r_{x,\pi(x)}) \]

**Optimal strategy:**
- **Expected reward**
  \[ E(\sum_{t=0}^{\infty} \gamma^t r_t) = E(r_{x_0,a_0}) + \gamma E(r_{x_1,a_1}) + \gamma^2 E(r_{x_2,a_2}) + \ldots + E(\gamma^k r_{x_k,a_k}) + \ldots \]
  \[ = E(r_{x_0,a_0}) + \gamma E(r_{x_1,a_1}) + \gamma^2 E(r_{x_2,a_2}) + \ldots + \gamma^k E(r_{x_k,a_k}) + \ldots \]
- **Optimizing the expected reward**
  \[ \max E(\sum_{t=0}^{\infty} \gamma^t r_t) = \max E(r_{x_0,a_0}) + \gamma \max E(r_{x_1,a_1}) + \ldots + \gamma^k \max E(r_{x_k,a_k}) + \ldots \]
  \[ = \max E(r_{x_0,a_0}) + \gamma \max E(r_{x_1,a_1}) + \ldots + \gamma^k \max E(r_{x_k,a_k}) + \ldots \]
  \[ = \max_{a_0} R(x_0, a_0) + \gamma \max_{a_1} R(x_1, a_1) + \ldots + \gamma^k \max_{a_k} R(x_k, a_k) + \ldots \]
- **Optimal strategy:** $\pi^*: X \rightarrow A$
  \[ \pi^*(x) = \arg \max_a R(x, a) \]
RL with immediate rewards

The optimal choice assumes we know the expected reward \( R(x, a) \)

- **Then:** \( \pi^*(x) = \arg \max_a R(x, a) \)

**Caveats**
- **We do not know the expected reward** \( R(x, a) \)
  - We need to estimate it using \( \tilde{R}(x, a) \) from interaction
- **We cannot determine the optimal policy if the estimate of the expected reward is not good**
  - We need to try also actions that look suboptimal \( \text{wrt} \) the current estimates of \( \tilde{R}(x, a) \)

---

**RL with immediate rewards**

- **Problem:** In the RL framework we do not know \( R(x, a) \)
  - The expected reward for performing action \( a \) at input \( x \)
- **Solution:**
  - For each input \( x \) try different actions \( a \)
  - Estimate \( R(x, a) \) using the average of observed rewards

\[
\tilde{R}(x, a) = \frac{1}{N_{x,a}} \sum_{i=1}^{N_{x,a}} r_{i,x,a}^a
\]

- Action choice \( \pi(x) = \arg \max \tilde{R}(x, a) \)
- Accuracy of the estimate: statistics (Hoeffding’s bound)

\[
P(\left| \tilde{R}(x, a) - R(x, a) \right| \geq \varepsilon) \leq \exp \left[ -\frac{2\varepsilon^2 N_{x,a}}{(r_{\max} - r_{\min})^2} \right] \leq \delta
\]

- Number of samples:

\[
N_{x,a} \geq \frac{(r_{\max} - r_{\min})^2}{2\varepsilon^2} \ln \frac{1}{\delta}
\]
RL with immediate rewards

• **On-line (stochastic approximation)**
  – An alternative way to estimate $R(x, a)$

• **Idea:**
  – choose action $a$ for input $x$ and observe a reward $r^{x,a}$
  – Update an estimate in every step $i$

$$
\tilde{R}(x, a)^{(i)} \leftarrow (1 - \alpha(i))\tilde{R}(x, a)^{(i-1)} + \alpha(i) r^{x,a}_i \ \
\alpha(i) - a learning rate
$$

• **Convergence property:** The approximation converges in the limit for an appropriate learning rate schedule.
• Assume: $\alpha(n(x, a))$ - is a learning rate for $n$th trial of $(x, a)$ pair
• Then the converge is assured if:
  1. $\sum_{i=1}^{\infty} \alpha(i) = \infty$
  2. $\sum_{i=1}^{\infty} \alpha(i)^2 < \infty$

---

RL with immediate rewards

• At any step in time $i$ during the experiment we have estimates of expected rewards for each $(coin, action)$ pair:
  $\tilde{R}(coin1, head)^{(i)}$
  $\tilde{R}(coin1, tail)^{(i)}$
  $\tilde{R}(coin2, head)^{(i)}$
  $\tilde{R}(coin2, tail)^{(i)}$
  $\tilde{R}(coin3, head)^{(i)}$
  $\tilde{R}(coin3, tail)^{(i)}$

• Assume the next coin to play in step $(i+1)$ is coin 2 and we pick head as our bet. Then we update $\tilde{R}(coin2, head)^{(i+1)}$ using the observed reward and one of the update strategy above, and keep the reward estimates for the remaining (coin, action) pairs unchanged, e.g. $\tilde{R}(coin2, tail)^{(i+1)} = \tilde{R}(coin2, tail)^{(i)}$
Exploration vs. Exploitation in RL

The learner actively interacts with the environment via actions:

- At the beginning the learner does not know anything about the environment
- It gradually gains the experience and learns how to react to the environment

**Dilemma (exploration-exploitation):**

- After some number of steps, should I select the best current choice (exploitation) or try to learn more about the environment (exploration)?
- **Exploitation** may involve the selection of a sub-optimal action and prevent the learning of the optimal choice
- **Exploration** may spend too much time on trying bad currently suboptimal actions

---

Exploration vs. Exploitation

- In the RL framework
  - the learner actively interacts with the environment and **chooses the action** to play for the current input \( x \)
  - Also at any point in time it has an estimate of \( \hat{R}(x, a) \) for any \( (input, action) \) pair
- **Dilemma for choosing the action to play for** \( x \):
  - Should the learner choose the current best choice of action (exploitation)
    \[
    \hat{\pi}(x) = \arg \max_{a \in A} \hat{R}(x, a)
    \]
  - Or choose some other action \( a \) which may help to improve its \( \hat{R}(x, a) \) estimate (exploration)
This dilemma is called **exploration/exploitation dilemma**
- **Different exploration/exploitation strategies exist**
**Exploration vs. Exploitation**

- **Uniform exploration:**
  - Uses exploration parameter $0 \leq \varepsilon \leq 1$
  - Choose the “current” best choice with probability $1 - \varepsilon$
  
  \[ \hat{r}(x) = \arg \max_{a \in A} \tilde{R}(x, a) \]
  
  - All other choices are selected with a uniform probability $\varepsilon / |A| - 1$

**Advantages:**
- Simple, easy to implement

**Disadvantages:**
- Exploration more appropriate at the beginning when we do not have good estimates of $\tilde{R}(x, a)$
- Exploitation more appropriate later when we have good estimates

**Exploration vs. Exploitation**

- **Boltzmann exploration**
  - The action is chosen randomly but proportionally to its current expected reward estimate
  - Can be tuned with a temperature parameter $T$ to promote exploration or exploitation

  - Probability of choosing action $a$
  
  \[ p(a \mid x) = \frac{\exp[\tilde{R}(x, a) / T]}{\sum_{a' \in A} \exp[\tilde{R}(x, a') / T]} \]

- **Effect of $T$:**
  - For high values of $T$, $p(a \mid x)$ is uniformly distributed for all actions
  - For low values of $T$, $p(a \mid x)$ of the action with the highest value of $\tilde{R}(x, a)$ is approaching 1