# Jack's car rental

Corrado Possieri

Machine and Reinforcement Learning in Control Applications

# Problem

Manage cars between two locations.

# Problem formulation

- Jack manages two locations for a car rental company.
- Each day, some customers arrive at each location to rent cars.
- If Jack has a car available, he rents it out and is credited \$10.
- Cars are available for renting the day after they are returned.
- Jack can move cars between the two locations overnight.
- The cost of moving a car is \$2.
- Each location is capable of accommodating 30 cars.

# Problem data

- Cars requested and returned at each location are Poisson random variables

$$\mathbb{P}[\text{cars} = n] = \frac{\lambda^n}{n!}\exp(-\lambda).$$

- $\lambda$ is $3$ and $4$ for rental requests.
- $\lambda$ is $3$ and $2$ for returns.
- Jack's foresight modeled with discount $\gamma = 0.9$.
- Jack can move up to $7$ cars between the two locations.

# Model

- We can model the process as an MDP.
- The state is the number of car at each location
  - is it a Markov state?
  - we have $\#1 \cdot \#2$ states.
- The action is the number of cars moved
  - we have $2\#c + 1$ actions.

# State update

1. Jack reintroduces car returned at previous day.
2. Jack rents available cars.
3. Jack moves cars between the two locations.

# Transition and reward

- Let $S$ and $A$ be the number of states and actions.
- Transition probabilities can be stored in a $S \times S \times A$ matrix $P$

$$P_{s,s',a} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

- Rewards can be stored in a $S \times A$ matrix $R$

$$R_{s,a} = \mathbb{E}[R_{t+1} | S_T = s, A_t = a].$$

# Transition probabilities

- Probability of returns do not depend on actions
  - define the return probability matrix $P_{\text{ret}}$

$$[P_{\text{ret}}]_{s,s'} = \mathbb{P}[S_{\text{after return}} = s' | S_t = s].$$

- Probability of rentals do not depend on actions
  - define the rental probability matrix $P_{\text{ren}}$

$$[P_{\text{ren}}]_{s,s'} = \mathbb{P}[S_{\text{after rental}} = s' | S_{\text{after return}} = s].$$

- Probability of movement depend on actions
  - define the movement probability matrix $P_{\text{mov}}$

$$[P_{\text{mov}}]_{s,s',a} = \mathbb{P}[S_{\text{after movement}} = s' | S_{\text{after rental}} = s, A_t = a].$$

- By the law of total probability

$$P = P_{\text{ret}} \cdot P_{\text{ren}} \cdot P_{\text{mov}}.$$

# Expected rewards

- Expected earning do not depend on action

$$\mathbb{E}\left[\text{earning}_{t+1} | S_{\text{after return}} = s\right] = \sum_r r\mathbb{P}\left[r | S_{\text{after return}} = s\right].$$

- By the law of total probability

$$\begin{aligned}&\mathbb{E}\left[\text{earning}_{t+1} | S_t = s\right]\\ &= \sum_{s'} \mathbb{P}[S_{\text{after return}} = s' | S_t = s] \sum_r r\mathbb{P}\left[r | S_{\text{after return}} = s'\right].\end{aligned}$$

- The expected reward is given by

$$R_{s,a} = [P_{\text{ret}} \cdot \text{earning}]_s - \text{cost}_a.$$

# PI and VI revisited

- Given a deterministic policy $\pi$, define
  - $$P^\pi = \begin{bmatrix} P_{1,1,\pi(1)} & P_{1,2,\pi(1)} & \cdots & P_{1,S,\pi(1)} \\ P_{2,1,\pi(2)} & P_{2,2,\pi(2)} & \cdots & P_{2,S,\pi(2)} \\ \vdots & \vdots & \ddots & \vdots \\ P_{S,1,\pi(S)} & P_{S,2,\pi(S)} & \cdots & P_{S,S,\pi(S)} \end{bmatrix};$$
  - $$R^\pi = \begin{bmatrix} R_{1,\pi(1)} \\ R_{2,\pi(2)} \\ \vdots \\ R_{S,\pi(S)} \end{bmatrix}$$
- Bellman expectation update can be rewritten as

$$v \leftarrow R^\pi + \gamma P^\pi v \qquad (v^\pi = (I - \gamma P^\pi) R^\pi).$$

- Bellman optimality update can be rewritten as

$$v \leftarrow \max_\pi \{R^\pi + \gamma P^\pi v\}$$

# Matrix Formulation

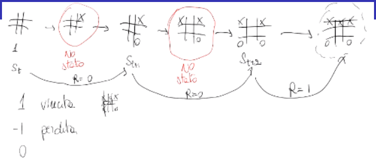
- Recall classical Bellman update

$$
\begin{aligned}
v(s) &\leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\left(r + \gamma v(s')\right) \\
&= \sum_{s',r} p(s',r|s,\pi(s))\left(r + \gamma v(s')\right) \\
&= \sum_{s',r} p(s',r|s,\pi(s)) r + \gamma \sum_{s',r} p(s',r|s,\pi(s)) v(s') \\
&= \sum_r r \sum_{s'} (s',r|s,\pi(s)) + \gamma \sum_{s'} v(s') \sum_r p(s',r|s,\pi(s)) \\
&= r(s,\pi(s)) + \gamma \sum_{s'} p(s'|s,\pi(s)) v(s').
\end{aligned}
$$

- A similar relation holds for Bellman optimality update

$$
v(s) \leftarrow \max_a \left\{ r(s,a) + \gamma \sum_{s'} p(s'|s,a) v(s') \right\}.
$$

# Assignment #2




- Write a code for PI (in class).
- Write a code for VI (in class).
- Model tic-tac-toe as an MDP
  - each board configuration is a state;
  - actions is where to place your mark;
  - the other player is playing at random;
  - use **afterstates**
    - evaluate bo[illegible] positions after the other p
  - reward $+1$ for winning and $-1$ for losing;
  - see Section 1.5 of textbook for some hints;
  - use VI and PI to determine optimal actions.