Speech and Language Processing

Lecture 6
Reinforcement Learning (2)
Bellman Equations

Information and Communications Engineering Course

Takahiro Shinozaki

Manabu Okumura

2024/10/2

Evaluations of Value Functions

In the previous lecture, we have defined value functions

- State-value function of MRP is the expected return starting from state s

$$v(s) = E[G_t|S_t = s] = \sum_{r_t, s_{t+1}, r_{t+1}, s_{t+2}, \cdots, s_\infty} P(r_t, s_{t+1}, r_{t+1}, s_{t+2}, \cdots, s_\infty | s_t = s)\, G_t$$

- State-value function of MDP is an expected return starting from state s and following the policy π

$$v^\pi(s) = E[G_t|S_t = s] = \sum_{a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, \cdots, s_\infty} P(a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, \cdots, s_\infty | s_t = s)\, G_t$$

- Action-value function of MDP is an expected return starting from state s, taking action a, and then following policy π

$$q_t^\pi(s, a) = E[G_t|S_t = s, A_t = a] = \sum_{r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, \cdots, s_\infty} P(r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, \cdots, s_\infty | S_t = s, A_t = a)\, G_t$$

We can evaluate these value functions by solving Bellman Equations

Evaluation of $v(s)$ of MRP

$$v(s) = \sum_{r_t, s_{t+1}, r_{t+1}, s_{t+2}, \cdots, s_\infty} P(r_t, s_{t+1}, r_{t+1}, s_{t+2}, \cdots, s_\infty | S_t = s) \sum_{k=0}^{\infty} \gamma^k r_{t+k}$$

$$= \sum_{r_t, s_{t+1}, \cdots, s_\infty} P(r_t, s_{t+1}, \cdots, s_\infty | S_t = s)\, \gamma^0 r_t + \sum_{r_t, s_{t+1}, \cdots, s_\infty} P(r_t, s_{t+1}, \cdots, s_\infty | S_t = s)\, \gamma^1 r_{t+1} + \cdots$$

$$= \sum_{r_t} P(r_t | S_t = s)\, \gamma^0 r_t + \sum_{r_{t+1}} P(r_{t+1} | S_t = s)\, \gamma^1 r_{t+1} + \sum_{r_{t+2}} P(r_{t+2} | S_t = s)\, \gamma^2 r_{t+2} + \cdots$$

$$= \gamma^0 \sum_{r_t} P(r_t | S_t = s)\, r_t + \gamma^1 \sum_{s_{t+1}} \sum_{r_{t+1}} P(S_{t+1} = s_{t+1} | S_t = s) P(r_{t+1} | s_{t+1})\, r_{t+1} + \cdots$$

$$= \bar{r}(s) + \gamma^1 \sum_{s'} P(S_{t+1} = s' | S_t = s) \bar{r}(s') + \gamma^2 \sum_{s'} P(S_{t+2} = s' | S_t = s) \bar{r}(s') + \cdots$$

($\bar{r}(s)$ is reward function)

Matrix Representation

Let $\boldsymbol{V} = \begin{bmatrix} v(s=1) \\ v(s=2) \\ \vdots \\ v(s=N) \end{bmatrix}, \boldsymbol{R} = \begin{bmatrix} \bar{r}(s=1) \\ \bar{r}(s=2) \\ \vdots \\ \bar{r}(s=N) \end{bmatrix}$

Because:

$$v(s) = \bar{r}(s) + \gamma^1 \sum_{s'} P(S_{t+1} = s' \,|S_t = s)\bar{r}(s') + \gamma^2 \sum_{s'} P(S_{t+2} = s' \,|S_t = s)\bar{r}(s') + \cdots$$

We have:

$$\boldsymbol{V} = \boldsymbol{R} + \gamma \boldsymbol{T}\boldsymbol{R} + \gamma^2 \boldsymbol{T}^2\boldsymbol{R} + \gamma^3 \boldsymbol{T}^3\boldsymbol{R} \cdots$$

(T is the transition matrix)

Vector version of geometric sequence

Bellman Equation for $v(s)$ of MRP

$$\boldsymbol{V} = \boldsymbol{R} + \gamma \boldsymbol{T}\boldsymbol{R} + \gamma^2 \boldsymbol{T}^2 \boldsymbol{R} + \gamma^3 \boldsymbol{T}^3 \boldsymbol{R} \cdots + \gamma^K \boldsymbol{T}^K \boldsymbol{R}$$

Multiply both sides by γT

Then subtract.

$$\gamma T \boldsymbol{V} = \quad \gamma \boldsymbol{T}\boldsymbol{R} + \gamma^2 \boldsymbol{T}^2 \boldsymbol{R} + \gamma^3 \boldsymbol{T}^3 \boldsymbol{R} \cdots + \gamma^K \boldsymbol{T}^K \boldsymbol{R} + \gamma^{K+1} \boldsymbol{T}^{K+1} \boldsymbol{R}$$

$$\boldsymbol{V} - \gamma \boldsymbol{T}\boldsymbol{V} = \boldsymbol{R} - \gamma^{K+1} \boldsymbol{T}^{K+1} \boldsymbol{R}$$

K→ ∞

$$\boldsymbol{V} = \boldsymbol{R} + \gamma \boldsymbol{T}\boldsymbol{V}$$

Bellman Equation
(in matrix form)

Bellman Equation for $v^{\pi}(s)$ of MDP

Obtained from the induced MRP

$$\begin{cases} T^{\pi}(s,s') = \sum_{a \in A} \pi(a|s) T(a,s,s') \\ R^{\pi}(s) = \sum_{a \in A} \pi(a|s) R(a,s) \end{cases}$$

Let $\boldsymbol{V^{\pi}} = \begin{bmatrix} v^{\pi}(s=1) \\ v^{\pi}(s=2) \\ \vdots \\ v^{\pi}(s=N) \end{bmatrix}$

$$\boldsymbol{V^{\pi}} = \boldsymbol{R^{\pi}} + \gamma T^{\pi} \boldsymbol{V^{\pi}}$$

Bellman Equation
(in matrix form)

Bellman Equation for $q^{\pi}(s,a)$ of MDP

Let

$$\bar{r}(a,s) = \bar{r}_t(a,s) = E[R_t = r | S_t = s, A_t = a] = \sum_r r\, P(R_t = r | S_t = s, A_t = a)$$

be a reward function of MDP

Bellman Equation

$$q^{\pi}(s,a) = \bar{r}(s,a) + \gamma \sum_{s'} T(a,s,s') \sum_{a'} \pi(a'|s')\, q^{\pi}(s',a')$$

Because $$v^{\pi}(s) = \sum_{a \in A} \pi(a|s) q^{\pi}(s,a)$$

We can also have $$q^{\pi}(s,a) = \bar{r}(a,s) + \gamma \sum_{s' \in S} T(a,s,s') v^{\pi}(s')$$

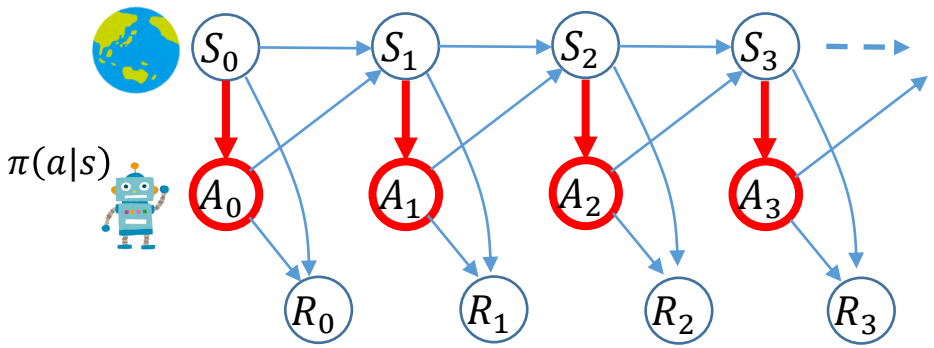
Direct Solution of Bellman Equation for $v^{\pi}(s)$

$$\boldsymbol{V} = \boldsymbol{R} + \gamma \boldsymbol{T}\boldsymbol{V} \qquad \boldsymbol{V} = \begin{bmatrix} v^{\pi}(s=1) \\ v^{\pi}(s=2) \\ \vdots \\ v^{\pi}(s=N) \end{bmatrix}, \boldsymbol{R} = \begin{bmatrix} \overline{r^{\pi}}(s=1) \\ \overline{r^{\pi}}(s=2) \\ \vdots \\ \overline{r^{\pi}}(s=N) \end{bmatrix}$$

$$\Rightarrow \quad (\boldsymbol{I} - \gamma \boldsymbol{T})\boldsymbol{V} = \boldsymbol{R}$$

$$\Rightarrow \quad \boldsymbol{V} = (\boldsymbol{I} - \gamma \boldsymbol{T})^{-1}\boldsymbol{R}$$

- The matrix size must be small to perform calculation (Calculation cost is $O(N^3)$)
- The agent must be able to directly access $P(s'|s,a)$ and $P(r|s,a)$



Iterative Solution by Dynamic Programming

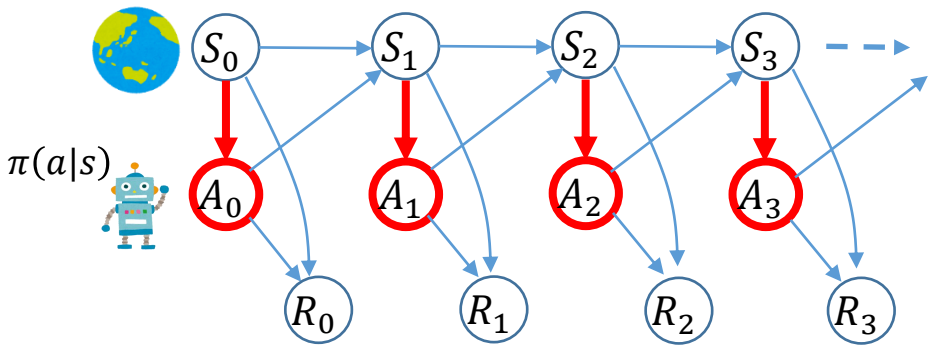
- Consider an update formula:

$$\boldsymbol{V}_{k+1} = \boldsymbol{R} + \gamma \boldsymbol{T}\boldsymbol{V}_k$$

- If it converges, the solution is obtained:

$$\boldsymbol{V} = \boldsymbol{V}_{k+1} = \boldsymbol{V}_k$$

- Calculation cost of one update is $O(N^2)$
- The agent must be able to directly access $P(s'|s,a)$ and $P(r|s,a)$



Proof of Convergence

$$\boldsymbol{V}_{k+1} = \boldsymbol{R} + \gamma \boldsymbol{T}\boldsymbol{V}_k$$

$$\boldsymbol{V} = \boldsymbol{R} + \gamma \boldsymbol{T}\boldsymbol{V}$$

Subtract

$$\boldsymbol{E}_{k+1} = \boldsymbol{V}_{k+1} - \boldsymbol{V} = \gamma \boldsymbol{T}(\boldsymbol{V}_k - \boldsymbol{V}) = \gamma \boldsymbol{T}\boldsymbol{E}_k = \gamma^{k+1}\boldsymbol{T}^{k+1}\boldsymbol{E}_0$$

$$\lim_{k\to\infty} \boldsymbol{E}_k = \boldsymbol{0} \qquad 0 < \gamma < 1$$

c.f. Jacobi method, Gauss-Seidel method

Evaluation by Sample Approximation

$$v_{k+1}^{\pi}(s) = \overline{r^{\pi}}(s) + \gamma \sum_{s'} T^{\pi}(s,s') v_k^{\pi}(s')$$

$$= \sum_r r \sum_{a \in A} \pi(a|s) P(r|s,a) + \gamma \sum_{s'} \sum_{a \in A} \pi(a|s) P(s'|s,a)\, v_k^{\pi}(s')$$

$$= \sum_r \sum_{s'} \sum_a \pi(a|s) P(r|s,a) P(s'|s,a) \{r + \gamma v_k^{\pi}(s')\}$$

$$= E[r + \gamma v_k^{\pi}(s')|s]$$

$$\approx \frac{1}{M} \sum_{m=1}^{M} r_m + \gamma v_k^{\pi}(s_m'), \qquad \langle r_m, s_m', a_m' \rangle \sim P(r, s', a'|s)$$

$$q_{\pi}(s,a)_{k+1} = E[r + \gamma q_{\pi}(s', a')_k | s, a]$$

$$\approx \frac{1}{M} \sum_{m=1}^{M} r_m + \gamma q_{\pi}(s_m', a_m')_k, \qquad \langle r_m, s_m', a_m' \rangle \sim P(r, s', a'|s, a)$$

Incremental Average Calculation

- Cumulative Average

Let $$CA_M = CA_{M-1} + \frac{1}{M}(x_M - CA_{M-1})$$

Then $$CA_M = \frac{x_1 + x_2 + \cdots + x_M}{M}$$

- Exponential Moving Average

Let $$EMA_M = EMA_{M-1} + \alpha(x_M - EMA_{M-1})$$

Then $$EMA_M = \frac{(1-\alpha)^{M-1}x_1 + (1-\alpha)^{M-2}x_2 + \cdots + (1-\alpha)^0 x_M}{(1-\alpha)^{M-1} + \cdots + (1-\alpha) + 1}$$

Put larger weights on recent samples and gradually forget old samples

Temporal Difference (TD) Method

Evaluate the value functions by sample approximation and incremental average calculation

$$v_{k+1}^{\pi}(s) \leftarrow v_k^{\pi}(s) + \alpha\left(r_m + \gamma v_k^{\pi}(s_m{}') - v_k^{\pi}(s)\right)$$

$$q^{\pi}(s,a)_{k+1} \leftarrow q^{\pi}(s,a)_k + \alpha(\underline{r_m + \gamma q^{\pi}(s_m', a_m')_k} - q^{\pi}(s,a)_k)$$

Referred to as TD target

Optimal Policy

- Partial ordering of policies

  policy π' is better than policy π if it gives higher value for all states

$$\pi \leq \pi' \text{ if } \forall s \; v^{\pi}(s) \leq v^{\pi'}(s)$$

- There exists an optimal policy π^* that is better than or equal to all other policies

$$\exists \pi^* \forall \pi \; \pi \leq \pi^*$$

- The goal of reinforcement learning is to find an optimal policy

Optimal Value Functions

- The optimal value functions are the maximum value function over all policies

$$v^*(s) = \max_\pi v^\pi(s)$$

$$q^*(s,a) = \max_\pi q^\pi(s,a)$$

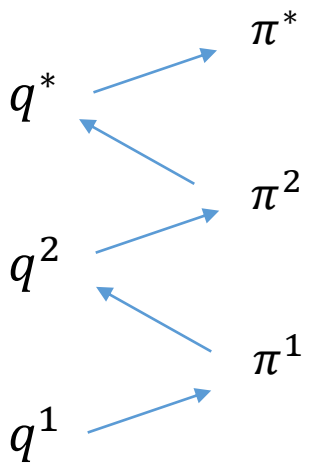
- When we have $q^*(s,a)$, we can obtain an optimal policy by:

$$\pi^*(a|s) = \begin{cases} 1 & if\ a = \operatorname{argmax}_a q^*(s,a) \\ 0 & otherwise \end{cases}$$

Exploration and Exploitation

Taking balance between search and utilize existing knowledge is important

- If we always seek unknown possibility, our average performance will be poor
- If we always act within existing knowledge, we have no chance of improvement



ϵ-Greedy Exploration

- Given an action-value function $Q(s, a)$, defines the policy $\pi(a|s)$ as:

$$\pi(a|s) = \begin{cases} \frac{\epsilon}{m} + 1 - \epsilon & \text{If } a = argmax_{a' \in A} Q(s, a') \\ \frac{\epsilon}{m} & \text{Otherwise} \end{cases}$$

SARSA: An Iterative Method to Obtain $q^*(s, a)$

Let $Q(s, a)$ be a table of $q(s, a)$ for s and a

Initialize $Q(s, a)$ and s.

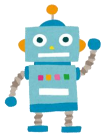
At each episode:

Choose an action a at state s based on a policy derived from $Q(s, a)$ using, e.g., ϵ-greedy

Execute the action a, and observe the reward r and transit to the next state s'

Choose an action a' at state s' based on a policy derived from $Q(s', a')$ using, e.g., ϵ-greedy

Repeat



s a r s' a'

- SARSA is model free
- SARSA is on-policy

$$Q(s, a) \leftarrow Q(s, a) + \alpha\big(r + \gamma Q(s', a') - Q(s, a)\big)$$
$$s \leftarrow s'$$
$$a \leftarrow a'$$

Q-Learning

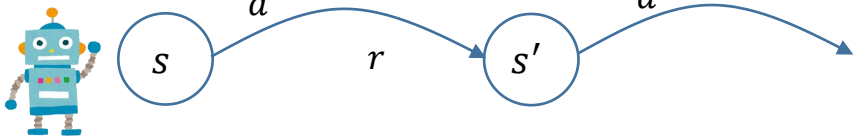
Initialize $Q(s, a)$ and s.

At each episode:

Choose an action a at state s based on a policy derived from $Q(s, a)$ using, e.g., ϵ-greedy

Repeat

Execute the action a, and observe the reward $r$and the next state s'



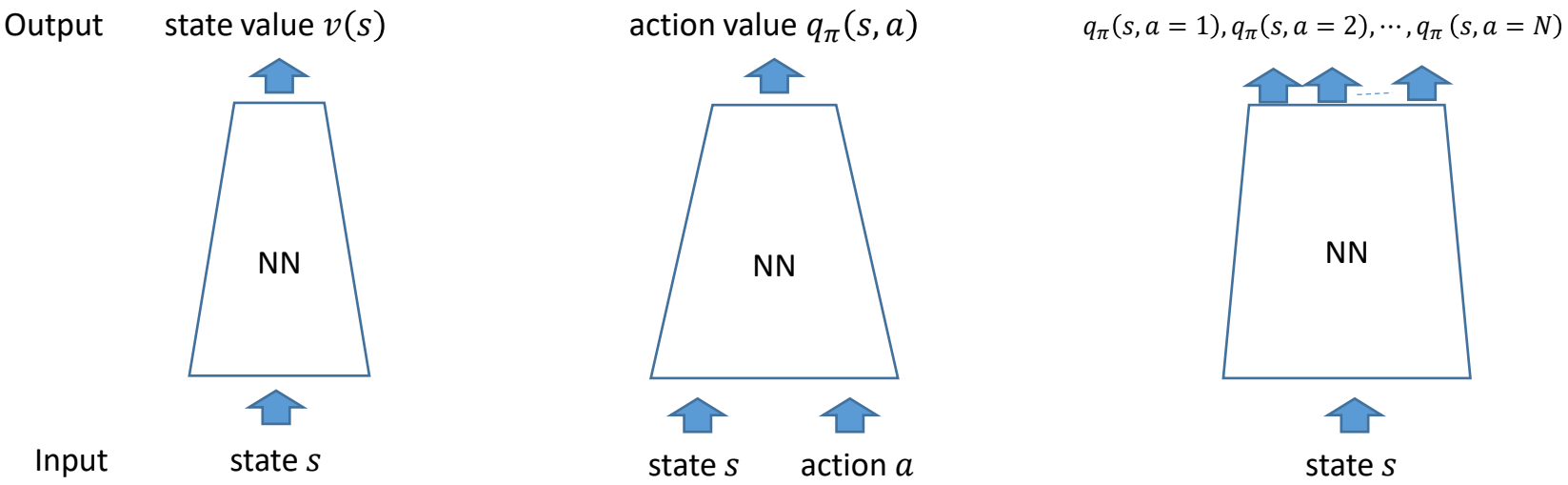
- Q-Learning is model free
- Q-Learning is off-policy

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right)$$

$$s \leftarrow s'$$

NN Implementation of Value Functions

By using NN instead of a table, we can handle larger state space



$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

⇨ Minimize $L = \left(r + \gamma \max_{a'} Q_{NN}(s',a') - Q_{NN}(s,a)\right)^2$

However, the learning is instable → DQN

Deep Q-Learning (DQN)

NN version of Q-learning + some heuristics

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

For episode = 1, M **do**

  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

  For $t = 1$,T **do**

    With probability ε select a random action a_t

    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

    Execute action a_t in emulator and observe reward r_t and image x_{t+1}

    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ

    Every C steps reset $\hat{Q} = Q$

  End For

End For

$$\left(r + \gamma \max_{a'} \widehat{Q_{\theta^-}}(s', a') - Q_\theta(s, a)\right)^2$$

☆Uses separated networks (prediction network Q_θ and target network $\widehat{Q_{\theta^-}}$)

☆ Uses experience replay

V. Mnih+, "Human-level control through deep reinforcement learning," Nature, 2015

Policy Gradient

- Directly parametrize the policy function and optimize policy performance objective

$$J(\theta) = \sum_s d^{\pi_\theta}(s)\, V^{\pi_\theta}(s) = \sum_s d^{\pi_\theta}(s) \sum_{a \in A} \pi(a|s) q^{\pi}(s,a)\,,$$

$$d^{\pi_\theta}(s) = \lim_{t \to \infty} P(s_t = s | s_0, \pi_\theta)$$

- Policy gradient theorem

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta log \pi_\theta(s,a) Q^{\pi_\theta}(s,a)]$$

R. Sutton+, "Policy Gradient Methods for Reinforcement Learning with Function Approximation," NISP, 1999
R. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," Machine Learning, 1992

REINFORCE

- Use return as an unbiased sample of $Q^{\pi_\theta}(s,a)$

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta log\pi_\theta(s,a)Q^{\pi_\theta}(s,a)] \approx E_{\pi_\theta}[\nabla_\theta log\pi_\theta(s,a)G_t]$$

$$\theta \leftarrow \theta + \alpha G_t \nabla_\theta \log\pi_\theta(s_t,a_t)$$

J. Schulman+, "High-Dimensional Continuous Control Using Generalized Advantage Estimation," ICLR 2016

Actor-Critic

- Parameterize $Q^{\pi_\theta}(s, a)$ and learns it together with policy π_θ

$$\nabla_\theta J(\theta) = E_{\pi_\theta}[\nabla_\theta log \pi_\theta(s, a) Q^{\pi_\theta}(s, a)] \approx E_{\pi_\theta}[\nabla_\theta log \pi_\theta(s, a) Q_w(s, a)]$$

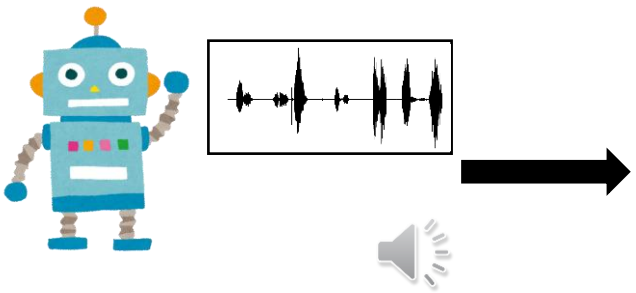
- Actor: π_θ (with parameters θ)
- Critic:$Q_w(s, a)$(with parameters w)

A Demo of Language Acquisition

- The robot has an intrinsic motivation to go to (0,0,0), i.e. homing instinct
- By correctly pronouncing a directional voice command, the robot can move one unit in that direction
- Initially, the robot has no language knowledge, and needs to learn the vocabulary of the voice commands and their meanings from scratch without relying on labeled data

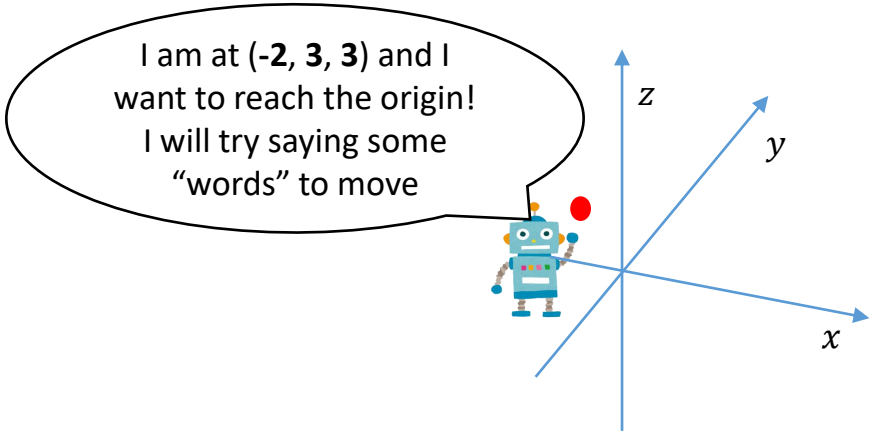
1. Observation based learning
(Unsupervised Learning)
Agent observes unsegmented spoken sound examples

2. Dialogue based learning
(Reinforcement Learning)
By pronouncing a command (up, down, left, right, forward, backward), agent can move one step to that direction at a time



Unsegmented sound sample

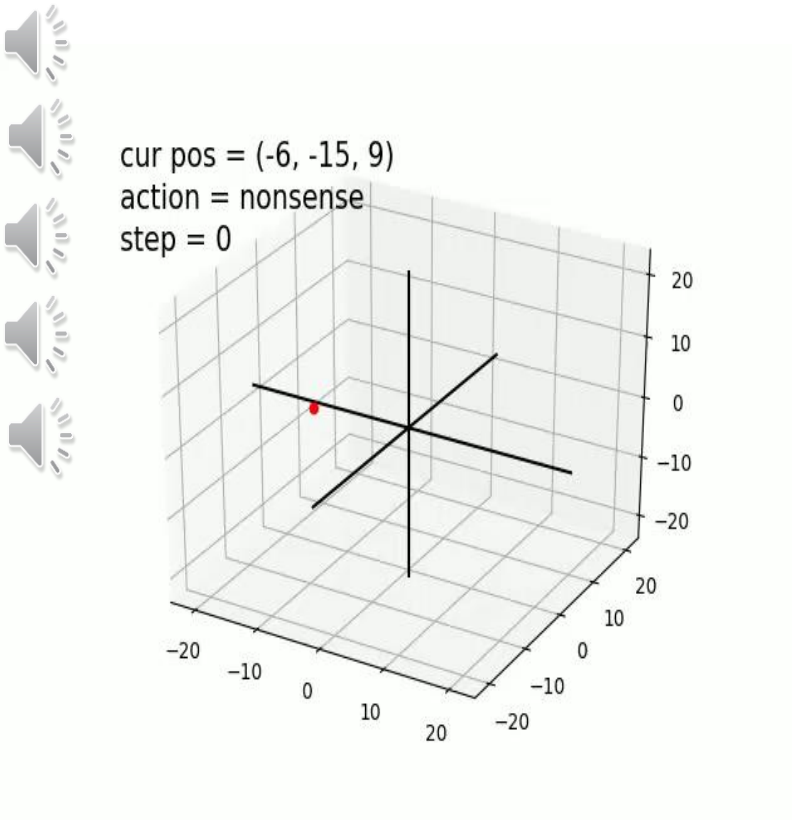
Initially, agent has no language knowledge



Agent is randomly positioned at a 3-D point (x, y, z) and wants to reach the origin

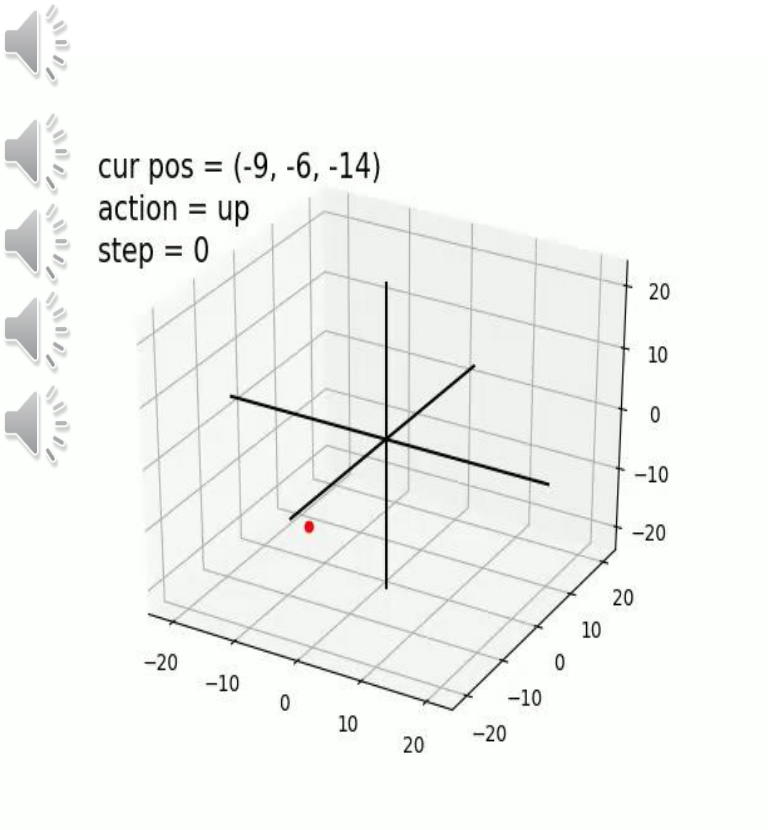
Language Acquisition by The Agent

- First five actions at episode 0



Moving process at episode 0

- First five actions at episode 5



Moving process at episode 5

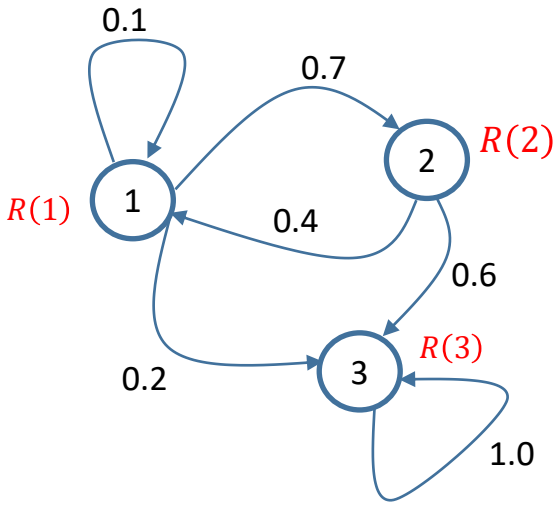
Exercise 6.1, 6.2

Consider the following Markov Reward Process.

$$S = \{1,2,3\}$$

$$I = [1.0 \quad 0.0 \quad 0.0]$$

$$T = \begin{bmatrix} 0.1 & 0.7 & 0.2 \\ 0.4 & 0 & 0.6 \\ 0 & 0 & 1.0 \end{bmatrix}$$



$$P(R_t = 0|S_t = 1) = 0.2, \quad P(R_t = 1|S_t = 1) = 0.8$$

$$P(R_t = 0|S_t = 2) = 0.2, \quad P(R_t = 1|S_t = 2) = 0.8$$

$$P(R_t = 0|S_t = 3) = 0.7, \quad P(R_t = 1|S_t = 3) = 0.3$$

$$\gamma = 0.8$$

6.1) Obtain $\bar{r}(s = 1)$

6.2) Obtain $v(s = 3)$