# Dynamic Programming

Hossein Hajiabolhassan

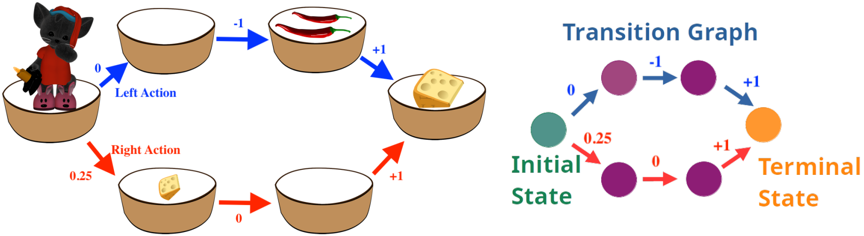
Department für Mathematik
und Informationstechnologie
Montanuniversität Leoben

November 22, 2022

# Mathematical Model of Environment: Markov Decision Process

# Transition Graph

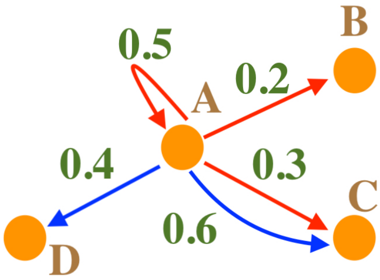
- By a model of the environment we mean anything that an agent can use to predict how the environment will respond to its actions.
- One can represent any environment and the way to interact with it by a transition graph. The rewards were written beside the edges.

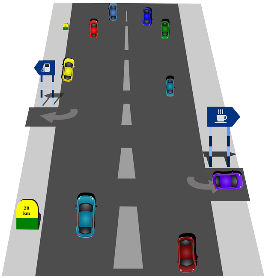



References 1, 2, 3, 4

# Stochastic Actions

- Stochastic actions can be represented by several edges. For instance, in the state $A$, there are two possible actions RED and BLUE. The transition probability were written beside the edges.



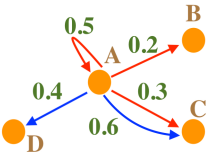

Reference

# Mathematical Model: Markov Decision Process

A Markov Decision Process (MDP) model $M = (S, A, T, R)$ is a 4-tuple:

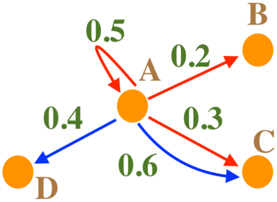
- State set: $s \in S$
- Transition function $T : S \times A \times S \to \mathbb{R}^{\geq 0}$
- Action set: $a \in A$
- A real valued reward function $R(s, a)$!

$T$ represents the probability of going from $s$ to $s'$ when executing action $a$: $\sum_{s' \in S} T(s, a, s') = 1$.



# Markov Decision Process

Assume the Markov Property: the effects of an action taken in a state depend only on that state and not on the prior history.



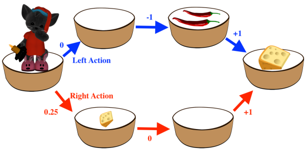

Question: Are real-world problems really Markovian? Everything is Markovian. Really all the physics (and chemistry and biology) most of us are interested in, in fact, is Markovian or effectively so (for more read Link: Everything is Markovian; nothing is Markovian)

# Policy

For MDPs, we aim to find an optimal policy $\pi : S \to A$.

- A policy $\pi$ gives an action for each state.
- An optimal policy is one that maximizes expected collected rewards.
- In general, policy can be
  - non-stationary (depend on the time)
  - stochastic: choose actions randomly
  - We just consider deterministic and stationary policies

Following right action is an optimal policy:



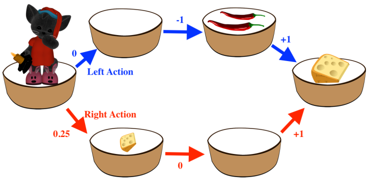

References 1, 2, 3, 4

# Policy

For MDPs, we aim to find an optimal policy $\pi : S \to A$.

## Utility of Following a Policy

Following a policy yields a (random) path! and one can collect the rewards of path.




References 1, 2, 3, 4

Following a policy $\pi$:

1. Determine the current state s
2. Execute action $\pi(s)$
3. Go to step 1

# Cumulative Rewards

## Definition (Cumulative Rewards)

In Episodic tasks, we can consider the cumulative rewards:

$$\sum_{i=0}^{t} R_{t+i}(S_{t+i}, A_{t+i})$$

Example: Playing Chess

- \$1 million for winning (the final action to reach the terminal state)
- -\$1 million for losing (the final action to reach the terminal state)
- -\$1 for any action except the final actions to reach the terminal state!

# Discounting Rewards

## Definition (Discounting Rewards)

In continuous and also episodic tasks, we can consider the discounting rewards:

$$\sum_{i=0}^{t} \gamma^i R_{t+i}(S_{t+i}, A_{t+i})$$

Why should we consider discounting reward rather than just add up?

- Mathematically more convenient formulation to deal with
- Bird in hand vs two in bush (e.g. monetary reward)': It's better to have a small, secured reward than the possibility of a bigger one
- Don't have infinite returns because of positive reward cycles in MDP
- It's reasonable to maximize the sum of rewards
- Can use $\gamma = 1$ if MDP is episodic

# Average Rewards

## Definition (Average Rewards)

In continuous tasks, we can consider the average rewards:

$$\frac{\sum_{i=0}^{t} R_{t+i}(S_{t+i}, A_{t+i})}{t}$$

It is recommended to consider average reward for continuous tasks! Why?

- Don't have infinite returns because of positive reward cycles in MDP

**Hereafter, we just consider cumulative and discounting rewards.**

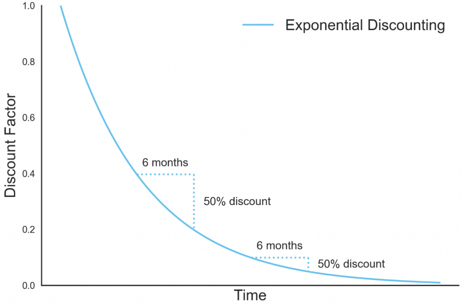
# Discounting Factor For Human

Valuing immediate money more than future money is a rational behavior known as discounting. Everybody has their own discount factor. In the early 1980s, psychologist George Ainslie discovered something peculiar about discount factor as shown in below. Reference


Reference

# Discounting Factor For Human

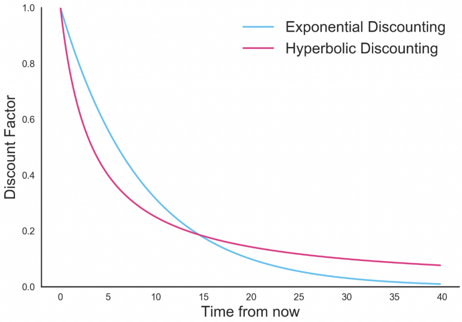
With an exponential curve, a dollar delayed by six months is always worth the same fixed fraction of a dollar at the baseline date, no matter what the baseline date it is. ▸ Reference




▸ Reference

# Discounting Factor For Human

In contrast to an exponential curve, humans tend to show a hyperbolic discount curve, which is considered irrational according to standard economic theory. Reference




Reference

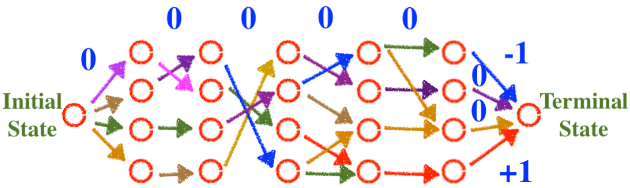
# The Problem of Delayed Reward

**Challenge**

The Problem of Delayed Reward

Goal: Select the optimal policy: maximise total future function reward

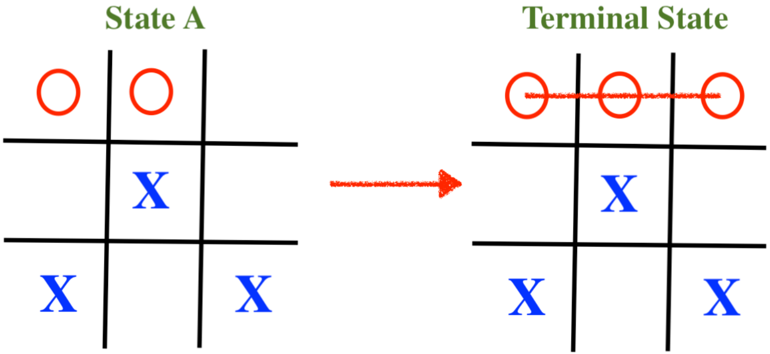
- Actions have long term consequences and Reward may be delayed.
- For instance in Chess, which action in long sequence is responsible for the win or loss?




The agent performs many actions and only receive **reward** or **punishment** at the **end** of episode, e.g., Games: Chess, Go, …

# Episodic Task

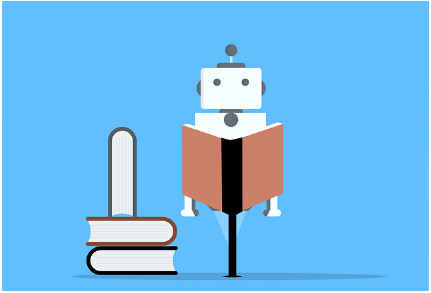
Episodic tasks are the tasks that have a terminal state (end). In reinforcement learning, episodes are considered agent-environment interactions from initial to terminal states.



# Continuous Task

In a continuous task, there is not a terminal state. Continuous tasks will never end.

For example, a personal assistant robot does not have a terminal state.

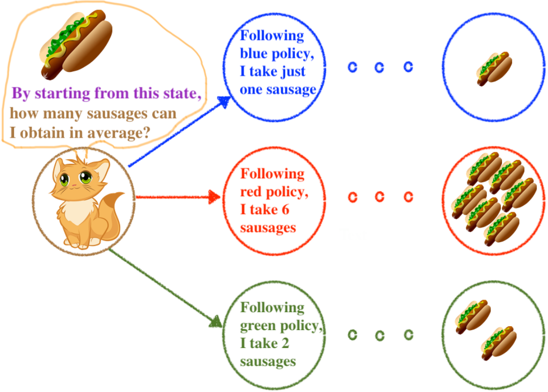


Reference

# Reinforcement Learning Algorithms

# Value of a State

In general, we seek to maximize the expected return as some specific function of the reward sequence. State values are a way to measure longer term benefits of being in a state.



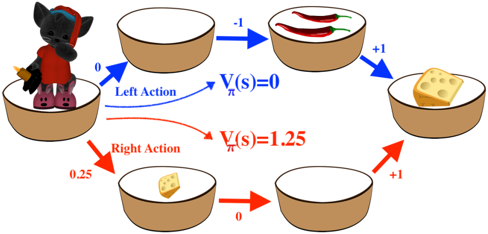

References 1, 2

The value function of a state $s$ under a policy $\pi$, denoted $v_\pi(s)$, is the expected return when starting in $s$ and following $\pi$ thereafter. Formally, we have

$$v_\pi(s) = E_\pi[Return|S_t = s]$$

Value function based on Cumulative Reward:

$$v_\pi(s) = E_\pi[\sum_{i=0}^{n} R_{t+i}|S_t = s]$$




References 1, 2, 3, 4

$$v_\pi(s) = E_\pi[R_t + R_{t+1} + R_{t+2} + \cdots |S_t = s]$$

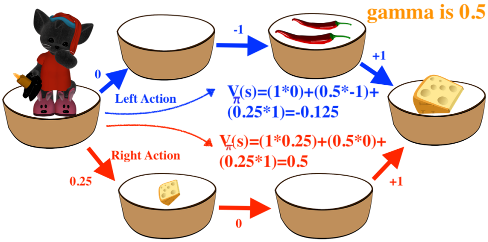
# Discounting Reward

The value function of a state $s$ under a policy $\pi$, denoted $v_\pi(s)$, is the expected return when starting in $s$ and following $\pi$ thereafter. Formally, we have

$$v_\pi(s) = E_\pi[Return|S_t = s]$$

Value function based on Discounting Reward:

$$v_\pi(s) = E_\pi[\sum_{i=0}^{n} \gamma^i R_{t+i}|S_t = s]$$




References 1, 2, 3, 4

If $\gamma = 0.5$, then

$$v_\pi(s) = E_\pi[0.5^0 R_t + 0.5^1 R_{t+1} + 0.5^2 R_{t+2} + \cdots |S_t = s]$$

# Value Function

The value function of a state $s$ under a policy $\pi$, denoted $v_\pi(s)$, is the expected return when starting in $s$ and following $\pi$ thereafter.

$$v_\pi(s) = E_\pi[\sum_{i=0}^{n} \gamma^i R_{t+i} | S_t = s]$$

If $\pi$ is deterministic, then for any state $s \in S$:

$$V_\pi(s) = R(s, \pi(s)) + \gamma \sum_{s'} T(s, \pi(s), s') V_\pi(s')$$

In view of definition of $v_\pi(.)$, we are naturally looking for an optimal policy $\pi^*$ where for any policy $\pi$ and state $s \in S$:

$$V_{\pi^*}(s) \geq V_\pi(s)$$

# Value Iteration Algorithm

**Goal**

Finding an optimal policy $\pi^*$

Algorithm: Consider a small value $0 < \epsilon < 1$ and set $\Delta = 0$
Assign a random value to each state and zero to terminal state
Loop (until $\Delta < \epsilon$):

- for each $s \in S$ do:
  1. $V_{Former}(s) = V(s)$
  2. Value Update: $V(s) = \underset{a \in A}{\text{argmax}} R(s,a) + \gamma \sum_{s'} T(s,a,s')V(s')$
  3. Set $\Delta = |V_{Former}(s) - V(s)|$

Return the Optimal Policy via Bellman Optimality Equation:

$$\pi^*(s) := \underset{a \in A}{\text{argmax}} R(s,a) + \gamma \sum_{s'} T(s,a,s')V(s')$$

# Policy Iteration Algorithm

**Goal**

Finding an optimal policy $\pi^*$

Algorithm: Consider a small value $0 < \epsilon < 1$ and set $\Delta = 0$

Assign a random value to each state and consider a random policy $\pi$

Loop (until $\Delta < \epsilon$):

- for each $s \in S$ do:
  1. $V_{Former}(s) = V(s)$
  2. Value Update: $V(s) = R(s, \pi(s)) + \gamma \sum_{s'} T(s, \pi(s), s') V_\pi(s')$
  3. Set $\Delta = |V_{Former}(s) - V(s)|$

Set $\pi^*(s) := \underset{a \in A}{\operatorname{argmax}} R(s, a) + \gamma \sum_{s'} T(s, a, s') V(s')$

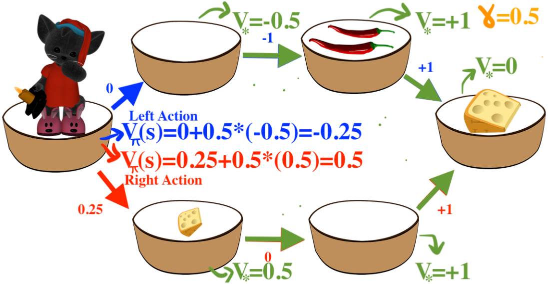
if $\pi^* \neq \pi$, set $\pi = \pi^*$ and run loop again, otherwise output $\pi^*$.

# Optimal Policy

## Theorem

*If one of the conditions holds, algorithm converges to optimal policy*

- $\gamma < 1$
- $\gamma = 1$ *and transition graph is acyclic*




References 1, 2, 3, 4

# Policy Iteration vs Value Iteration

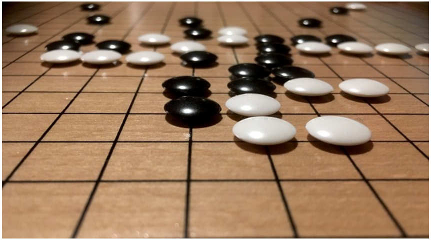
## Policy Iteration vs Value Iteration

1. Value Iteration:
   Pros: each iteration is very computationally efficient.
   Cons: convergence is only asymptotic.
2. Policy Iteration:
   Pros: converge in a finite number of iterations (often small in practice).
   Cons: each iteration requires a full policy evaluation and it might be expensive.

# Generalization

**Challenge**

How is it possible to find the optimal policy when the environment is huge or when there are insufficient samples?



Reference

# Machine Learning: Two Extremes

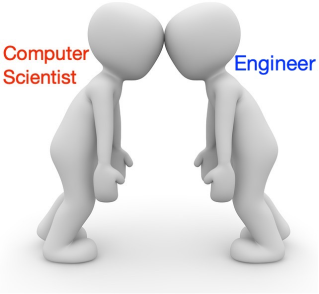
Ultimate Goal: For given problem, find algorithm that works well in practice (performance, efficiency), has favorable theoretical guarantees.

Engineer: My algorithm works well and it is computationally efficient but yours doesn't.

Computer Scientist: Your algorithm works for a special case. Prove that it works generally!

Engineer: Your algorithms work with some unusable assumptions or for small environment.

Computer Scientist: You can never introduce some general and efficient algorithm without my help.




Reference

# Acknowledgement

Thanks for your attention!