ECE 1508: Reinforcement Learning

Chapter 1: Introduction

Ali Bereyhi

ali.bereyhi@utoronto.ca

Department of Electrical and Computer Engineering

University of Toronto

Summer 2024

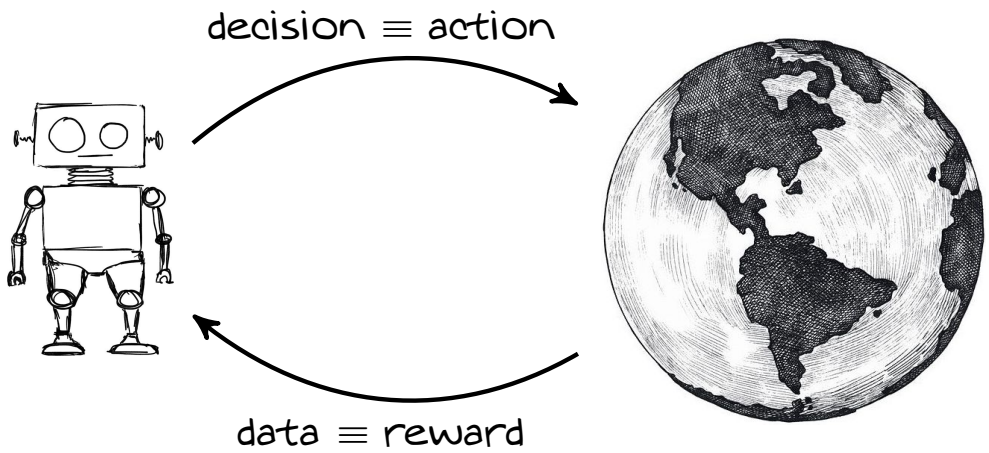
What is Reinforcement Learning?

David Silver from *Deep Mind* calls Reinforcement Learning

the art of decision making

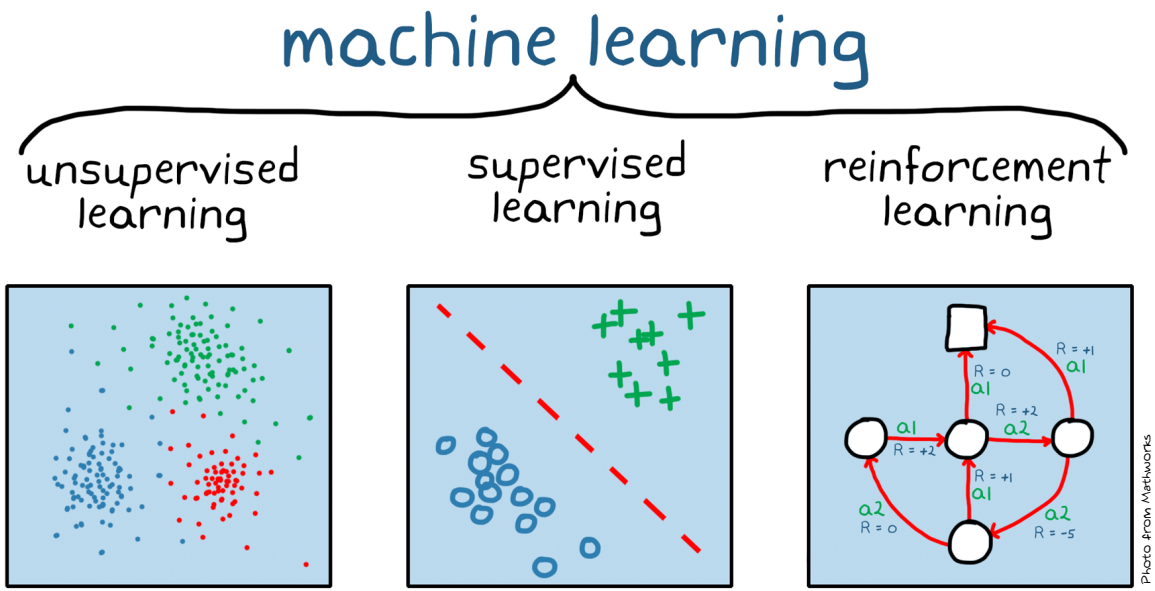
Of course *David Silver* can say it! But, for us as beginners it's better to say

it is all about learning how to interact with environment



We learn many things in our life by this approach

What is Reinforcement Learning?



Reinforcement learning is still a learning problem since

we still learn patterns or behaviors from data $\equiv$ observations

But, it has some fundamental differences to classical learning problems

What is Reinforcement Learning?

Reinforcement learning is *not supervised*

- *We don't collect data with labels*
- *We only see some rewards time to time*
  - *We need to use them to adjust our behavior*

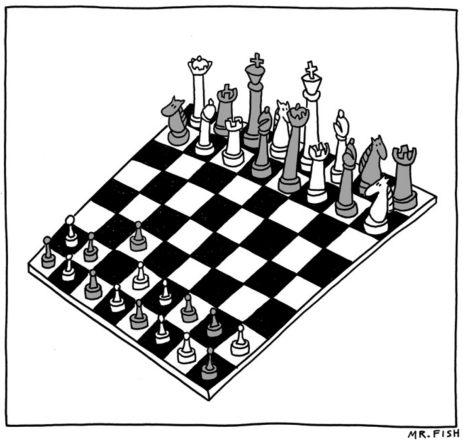
Reinforcement learning does *not* have *conventional dataset*

- *Each time we decide for an action we get a new feedback*
- *This feedback contributes to our learning* $\equiv$ *a new data-point*
  - *Our dataset is constructed through time*
  - *Our decisions affect the data we collect*

In reinforcement learning *time really matters*

- *We should act to see the next data*
- *It's also different from classical sequential data*
  - *In classical form, we see the whole sequence before we start with training*

What is Reinforcement Learning?



Best example of a reinforcement setting is a game

- *In games, we start from noting*
  - ↳ *We know just a bunch of rules*
- *We decide for a move ≡ an action*
  - ↳ *We then wait for the other side to move*
- *We decide for next move based on our observation*

- *Reinforcement learning is not supervised*
  - ↳ *We can't label a move as good or bad!*
- *Reinforcement learning does not have conventional dataset*
  - ↳ *Data is what we see through this game and maybe our earlier plays*
  - ↳ *Our current move impacts future moves in the game*
- *In reinforcement learning time really matters*
  - ↳ *We can only decide for the next move once we've seen the opponent's move*

Reinforcement Learning: *Achievements*

Reinforcement Learning has been hugely *revisited* in recent years

Alpha-Zero could beat world champions

take a look at this video where you could laso meet David Silver ☺

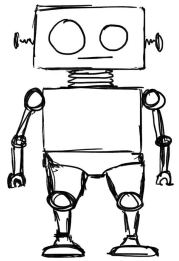
We are getting to there at some point, but first let's go back to 1952 and look at Herbert Robbins' multi-armed bandit problem which is pretty much

the most classic reinforcement learning problem

This helps us develop some intuition

Multi-armed Bandit

There are lots of variants! Let's start with our silly one: *you have developed a programming robot as your course project*

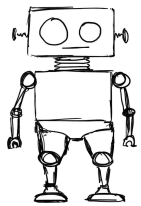


This robot can program in almost all programming languages. You now come up with a brilliant idea

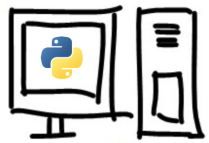
You plan to rent it daily to make some money!

Multi-armed Bandit

Your robot finds two options



Company A



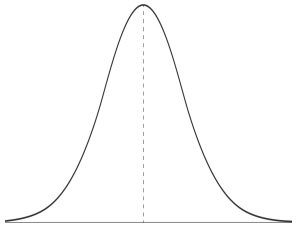
Company B

- *Company A which codes in Java*
- *Company B which codes in Python*

But none of these companies has a fixed daily payment

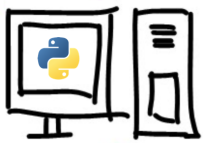
their payments is randomly changing every day

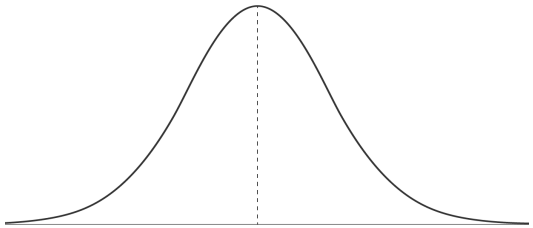
Multi-armed Bandit



Company A pays rent R_A which is Gaussian random variable

- *It has mean $\mu_A = 600$* ⇝ it pays in average $600 per day
- *Its standard deviation is $\sigma_A = 100$* ⇝ some days it may even pay $300





Company B pays rent R_B which is Gaussian random variable

- *It has mean $\mu_B = 400$* ⇝ it pays in average $400 per day
- *Its standard deviation is $\sigma_B = 200$* ⇝ some days it may even pay $1000

Multi-armed Bandit: *Known Distributions*

Obviously, the money we make depends on our *renting policy*

Policy

$\pi(t)$ *is the renting policy which specifies the selected company on day* t

$$\pi(t) = \begin{cases} A & \text{if we select Company A on day } t \\ B & \text{if we select Company B on day } t \end{cases}$$

Now, the main question is

What is the optimal renting policy?

To answer this question *we need to define what we mean by optimal*

Multi-armed Bandit: *Goal*

+ *Companies have stochastic payments! How can we define optimality?*
– Well, let's try looking at expected return per day

Average Return

Say we rent out for a period of T days, the average return is

$$G_T = \mathbb{E}\left\{\frac{1}{T}\sum_{t=1}^{T} R_t\right\}$$

where R_t is the rent paid on day t

We now set our goal to maximize the average return

optimal policy $\equiv$ *maximum average return*

Review: *Expectation and Conditional Expectation*

We need to recall expectation both marginal and conditional

Expectation

Let's say X and Y are two random variables; we need to know,

- *How to compute marginal expectation of X*

$$\mathbb{E}\{X\}$$

- *How to compute expectation of X conditional to Y*

$$\mathbb{E}\{X|Y=y\}$$

If you don't remember clearly

Please look at the blackboard!

Multi-armed Bandit: *Known Distributions*

+ *But, isn't the answer obvious?!*
– If *we know the distributions yes!*

With known distributions, we could simply write

$$G_T = \mathbb{E}\left\{\frac{1}{T}\sum_{t=1}^{T} R_t\right\} = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left\{R_t\right\} = \frac{T_A\mu_A + T_B\mu_B}{T}$$

where T_A and T_B are defined to be

- T_A *is the number of days we rent to Company* A
- T_B *is the number of days we rent to Company* B

We can obviously say that $T_A, T_B \leqslant T$ and

$$T_A = T - T_B$$

Multi-armed Bandit: *Known Distributions*

The return is

$$G_T = \frac{T_A \mu_A + T_B \mu_B}{T}$$

We can now write

$$G_T = \frac{(T - T_B)\,\mu_A + T_B \mu_B}{T} = \left(1 - \frac{T_B}{T}\right)\mu_A + \frac{T_B}{T}\mu_B$$
$$= \mu_A - \frac{T_B}{T}\left(\mu_A - \mu_B\right)$$

Recall that $\mu_A = 600$ *and* $\mu_B = 400$*, so we have*

$$G_T = 600 - 200\frac{T_B}{T}$$

Multi-armed Bandit: *Known Distributions*

The return is

$$G_T = 600 - 200\frac{T_B}{T}$$

We know that

always rent to A $\leftsquigarrow$ $0 \leqslant \frac{T_B}{T} \leqslant 1$ $\rightsquigarrow$ always rent to B

Therefore, optimal return is when $T_B/T = 0$

$$G_T^\star = 600$$

Multi-armed Bandit: *Known Distributions*

Optimal return is $G_T^\star = 600$

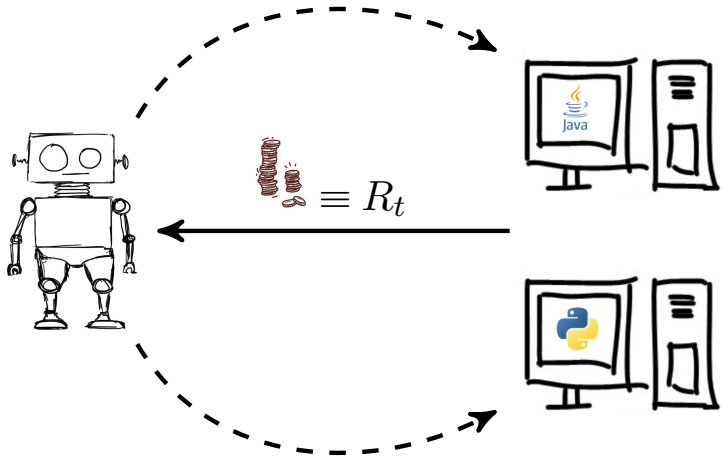
The optimal policy to achieve this return is to

always rent to Company A $\leftrightsquigarrow \pi^\star(t) = A$ *for all* t

---

+ *This was pretty obvious! Then, what is special about this problem?*
– The problem is that *we don't know the distributions in practice!*

Multi-armed Bandit



In practice, we can only *observe* *payments done each day by the companies*

we should decide the renting policy based on our observations

+ *OK! It seems hard to find an optimal policy!*
– Well! Let's take a look

Multi-armed Bandit: *Exploring Policy*

We start by a dumb policy: *we flip a uniform coin every day to select company*

$$\pi(t) = \begin{cases} A & \text{with Probability } 0.5 \\ B & \text{with Probability } 0.5 \end{cases}$$

Let's compute the return in this case

we denote it as $G_T^{(\mathrm{r})}$ *as it's a random policy*

Multi-armed Bandit: *Exploring Policy*

We have in this case

$$
\begin{aligned}
G_T^{(\mathrm{r})} &= \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left\{R_t\right\} = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\pi(t)}\left\{\mathbb{E}_{R_t}\left\{R_t|\pi\left(t\right)\right\}\right\} \\
&= \frac{1}{T}\sum_{t=1}^{T}0.5\mathbb{E}_{R_t}\left\{R_t|\pi\left(t\right)=A\right\}+0.5\mathbb{E}_{R_t}\left\{R_t|\pi\left(t\right)=B\right\} \\
&= \frac{1}{T}\sum_{t=1}^{T}0.5\mu_A+0.5\mu_B = 0.5\mu_A+0.5\mu_B = 500
\end{aligned}
$$

Now comparing to the maximal return we could get, we achieve

$$
\text{regret} \rightsquigarrow \rho^{(\mathrm{r})} = G_T^{\star} - G_T^{(\mathrm{r})} = 600 - 500 = 100
$$

less return!

Multi-armed Bandit: *Exploring-Exploiting Policy*

+ *But, can we get closer to the maximal return?*
– Sure! We should try to *learn the behavior of the companies*

In the previous approach, *we were only exploring the companies*

↳ *Maybe, we should exploit our exploration*
  ↳ *We get a rough idea about companies, once we get paid by them*
    ↳ *We could try both companies*
  ↳ *If we conclude that one company pays more*
    ↳ *We could stick to that company for the rest of the days*
↳ *This sounds like a learning procedure*
  ↳ *We try to learn the distribution of payments*

Multi-armed Bandit: *Exploring-Exploiting Policy*

Let's make another policy: *we try each company once, and stick to the one with who pays higher, i.e.,* $\pi(1) = A$, $\pi(2) = B$, *and*

$$\pi(t) = \begin{cases} A & \text{if } R_1 > R_2 \\ B & \text{if } R_1 \leqslant R_2 \end{cases}$$

Let's compute the return in this case

we denote it as $G_T^{(1)}$ *as we explore only once*

Review: *Properties of Gaussian Variables*

We need to recall Gaussian distribution and its properties

Expectation

We need to know,

- *What* $X \sim \mathcal{N}\left(\mu, \sigma^2\right)$ *means*
- *What Q-function is*
- *What the distribution of sum of Gaussian variables is*

If you don't remember clearly

Please look at the blackboard!

Multi-armed Bandit: *Exploring-Exploiting Policy*

Let's start with finding the return

$$
\begin{aligned}
G_T^{(1)} &= \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left\{R_t\right\} \\
&= \frac{1}{T}\left(\mathbb{E}\left\{R_1\right\} + \mathbb{E}\left\{R_2\right\} + \sum_{t=3}^{T}\mathbb{E}_{\pi(t)}\left\{\mathbb{E}_{R_t}\left\{R_t|\pi\left(t\right)\right\}\right\}\right) \\
&= \frac{1}{T}\left(\mu_A + \mu_B + \sum_{t=3}^{T}\Pr\left\{\pi\left(t\right)=A\right\}\mu_A + \left(1-\Pr\left\{\pi\left(t\right)=A\right\}\right)\mu_B\right)
\end{aligned}
$$

Let's define $\Pr\left\{\pi\left(t\right)=A\right\}=p_1$*; then, we can write*

$$
G_T^{(1)} = \frac{1}{T}\left(\mu_A + \mu_B + \left(T-2\right)\left[p_1\mu_A + \left(1-p_1\right)\mu_B\right]\right)
$$

Multi-armed Bandit: *Exploring-Exploiting Policy*

We have in this case

$$p_1 = \Pr\{\pi(t) = A\} = \Pr\{R_1 > R_2\} = \Pr\{R_1 - R_2 > 0\}$$

Let's look at the random variable $\Delta = R_1 - R_2$*:* R_1 *and* R_2 *are Gaussian*

- Δ *is also a Gaussian variable*
  - *It's mean is*

$$\mu_\Delta = \mathbb{E}\{R_1 - R_2\} = \mu_A - \mu_B = 200$$

  - *It's standard variation is*

$$\sigma_\Delta = \sqrt{\sigma_A^2 + \sigma_B^2} = \sqrt{50000} = 223.6$$

Well, p_1 is readily computed

$$p_1 = \Pr\{\Delta > 0\} = \mathrm{Q}\left(-\frac{200}{223.6}\right) = \mathrm{Q}(-0.89) \approx 0.81$$

Multi-armed Bandit: *Exploring-Exploiting Policy*

The average return is hence given by

$$
\begin{aligned}
G_T^{(1)} &= \frac{\mu_A + \mu_B}{T} + \left(1 - \frac{2}{T}\right)(0.81\mu_A + 0.19\mu_B) \\
&= (0.81\mu_A + 0.19\mu_B) - \frac{0.62}{T}(\mu_A - \mu_B) \\
&= 562 - \frac{124}{T}
\end{aligned}
$$

Comparing to maximal average return, we have

$$
\rho^{(1)} = G_T^\star - G_T^{(1)} = 38 + \frac{124}{T}
$$

which can be much less than $\rho^{(\mathrm{r})} = 100$ *if* T *is large enough!*

Multi-armed Bandit: *Exploring-Exploiting Policy*

There is however an obvious problem with this approach

Though better in average, it's not that reliable!

Reliability Issue

This would not be reliable in general: we could get for one random sample $R_1 \leqslant R_2$ even though we have $\mu_A > \mu_B$!

+ *Do you think we can make it more reliable?*
– Sure! Let's explore a bit more

Multi-armed Bandit: *Exploring-Exploiting Policy*

Let's make the policy more reliable: *we try each company d days and stick to the one with higher sum payments, i.e., we set $\pi(t) = A$ for $t = 1, \ldots, d$ and then compute a parameter*

$$S_A = \sum_{t=1}^{d} R_t$$

We then set $\pi(t) = B$ for $t = d+1, \ldots, 2d$ and compute

$$S_B = \sum_{t=d+1}^{2d} R_t$$

We finally set for $t > 2d$

$$\pi(t) = \begin{cases} A & \text{if } S_A > S_B \\ B & \text{if } S_A \leqslant S_B \end{cases}$$

Multi-armed Bandit: *Exploring-Exploiting Policy*

We can follow the same lines of calculation

$$
\begin{aligned}
G_T^{(d)} &= \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left\{R_t\right\} \\
&= \frac{1}{T}\left(\sum_{t=1}^{d}\mathbb{E}\left\{R_t\right\} + \sum_{t=d+1}^{2d}\mathbb{E}\left\{R_t\right\} + \sum_{t=2d+1}^{T}\mathbb{E}_{\pi(t)}\left\{\mathbb{E}_{R_t}\left\{R_t|\pi\left(t\right)\right\}\right\}\right) \\
&= \frac{1}{T}\left(d\mu_A + d\mu_B + \sum_{t=2d+1}^{T}\Pr\left\{\pi\left(t\right)=A\right\}\mu_A + \left(1-\Pr\left\{\pi\left(t\right)=A\right\}\right)\mu_B\right)
\end{aligned}
$$

We now define $\Pr\left\{\pi\left(t\right)=A\right\}=p_d$ *for* $t>2d$ *and write*

$$
G_T^{(d)} = \frac{1}{T}\left(d\left[\mu_A+\mu_B\right] + \left(T-2d\right)\left[p_d\mu_A + \left(1-p_d\right)\mu_B\right]\right)
$$

Multi-armed Bandit: *Exploring-Exploiting Policy*

We have in this case

$$p_d = \Pr\{\pi(t) = A\} = \Pr\{S_A > S_B\} = \Pr\{\Delta > 0\}$$

where we define

$$\Delta = S_A - S_B$$

- S_A *is sum of Gaussian variables* $\rightsquigarrow$ *it's Gaussian*
  - ↳ *It's mean is*

$$\mathbb{E}\{S_A\} = S_A = \sum_{t=1}^{d} \mathbb{E}\{R_t\} = d\mu_A$$

  - ↳ *It's standard variation is*

$$\sqrt{\mathbb{E}\left\{(S_A - d\mu_A)^2\right\}} = \sqrt{d\sigma_A^2} = \sigma_A\sqrt{d}$$

Multi-armed Bandit: *Exploring-Exploiting Policy*

We have in this case

$$p_d = \Pr\{\pi(t) = A\} = \Pr\{S_A > S_B\} = \Pr\{\Delta > 0\}$$

where we define

$$\Delta = S_A - S_B$$

- *S_B is sum of Gaussian variables $\rightsquigarrow$ it's Gaussian*
  - ↳ *It's mean is*

$$\mathbb{E}\{S_B\} = S_B = \sum_{t=d+1}^{2d} \mathbb{E}\{R_t\} = d\mu_B$$

  - ↳ *It's standard variation is*

$$\sqrt{\mathbb{E}\left\{(S_B - d\mu_B)^2\right\}} = \sqrt{d\sigma_B^2} = \sigma_B\sqrt{d}$$

Multi-armed Bandit: *Exploring-Exploiting Policy*

Let's look at the random variable $\Delta = S_A - S_B$: S_A *and* S_B *are Gaussian*

↳ Δ *is also a Gaussian variable*

↳ *It's mean is*

$$\mu_\Delta = \mathbb{E}\{S_A - S_B\} = d(\mu_A - \mu_B) = 200d$$

↳ *It's standard variation is*

$$\sigma_\Delta = \sqrt{\sigma_A^2 + \sigma_B^2}\sqrt{d} = 223.6\sqrt{d}$$

Well, p_d is readily computed

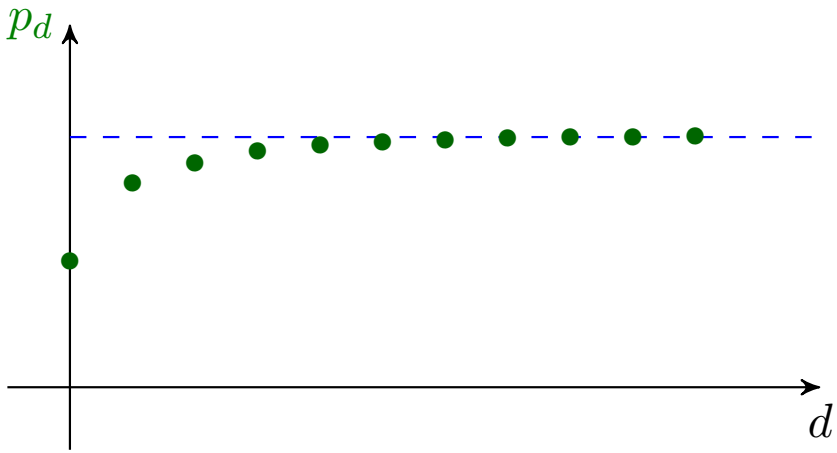
$$p_d = \Pr\{\Delta > 0\} = \mathrm{Q}\left(-\frac{200\sqrt{d}}{223.6}\right) = \mathrm{Q}\left(-0.89\sqrt{d}\right)$$

Multi-armed Bandit: *Exploring-Exploiting Policy*

The average return is hence given by

$$G_T^{(d)} = p_d \mu_A + (1 - p_d)\, \mu_B - \frac{2d}{T} (p_d - 0.5)\, (\mu_A - \mu_B)$$

where our p_d *quickly converges to one*



So we could say, for $d > 5$ *we have*

$$G_T^{(d)} \approx \mu_A - \frac{d}{T} (\mu_A - \mu_B)$$

Multi-armed Bandit: *Exploring-Exploiting Policy*

For slightly large d, we have

$$G_T^{(d)} \approx 600 - \frac{200d}{T}$$

So, we could conclude that

$$\rho^{(d)} \approx G_T^\star - G_T^{(d)} = \frac{200d}{T}$$

If we rent out for some long time, i.e., $T \gg d$; then, we eventually get to the optimal return

$$\lim_{T \to \infty} \rho^{(d)} = 0$$

Key Task: *Decision Making*

The key task in this problem was *decision making*

- *We train our robot to decide for best employer*
  - ↳ *This robot is referred to as an agent*
- *Everything would have been easy if we already had all data available*
  - ↳ *Before starting, we had lots of payment samples*
    - ↳ *We could then find out the payment distribution*
- *We are observing data while we are making decisions*
  - ↳ *We find out about payments after working there*
  - ↳ *We are interacting with an environment*
    - ↳ *The two companies in our example are the environment*

This is a simple example of a reinforcement learning problem

Let's introduce it and break its components down

Defining RL Problem

Let's make an agreement

$$\textit{Reinforcement Learning} \equiv \textit{RL}$$

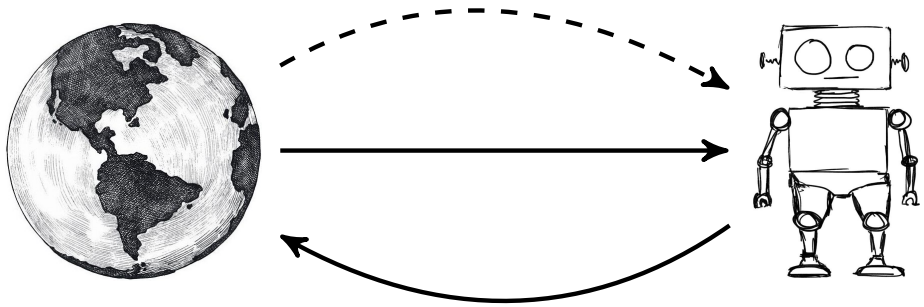
In each RL problem, we have three main components

1. *Agent who is the one who acts*
2. *Environment which responds to the agent's actoins*
3. *Interaction means which communicate actions and responds*

Let's mathematically model each component

Problem Formulation

The agent is interacting with the envronment

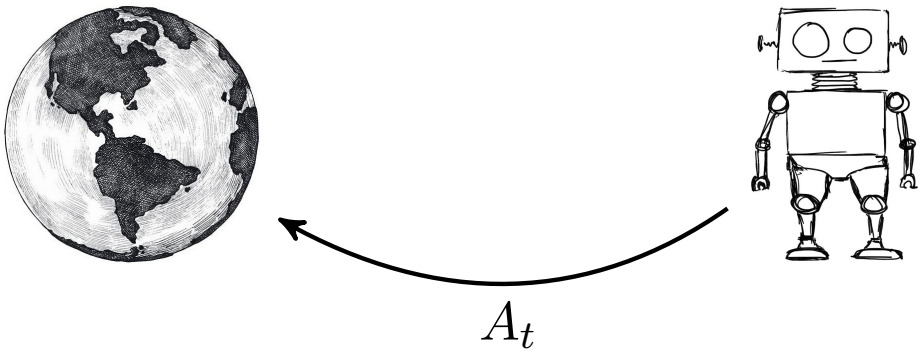


+ *How is exactly this interaction?*
- It happens by three means

  action, *observation* and *reward*

  Let's break it down

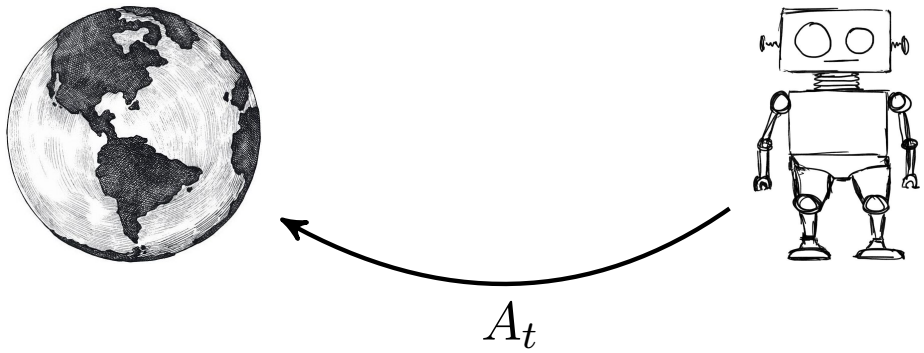
Problem Formulation: *Action*



At time t, agents selects an action from a set of possible actions

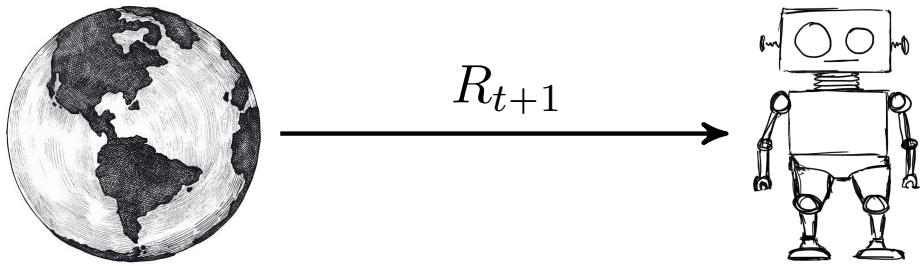
$$A_t \in \mathbb{A} = \{a^1, \ldots, a^M\}$$

Problem Formulation: *Action*



+ *Is this set always discrete?*
– *Not necessarily!* But, we may assume so for now ☺
+ *Is it always constant through time? Can't we get more or less options to act as time passes?*
– *Again: Not necessarily!* But, we may assume so for now ☺

Problem Formulation: *Reward*



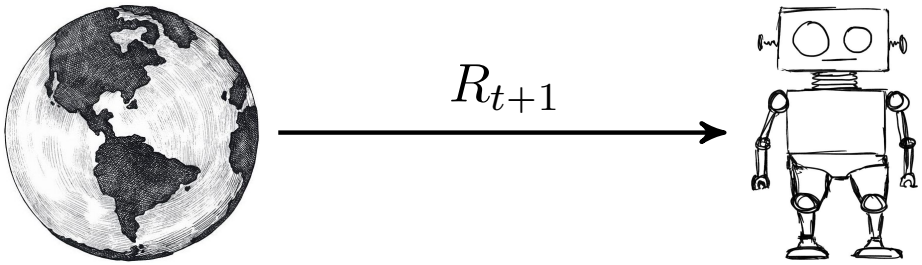
Once acted, the agent gets some reward from the environment

$$R_{t+1} \in \{r^1, \ldots, r^L\}$$

We index by $t+1$, as agent gets the reward in the next time step

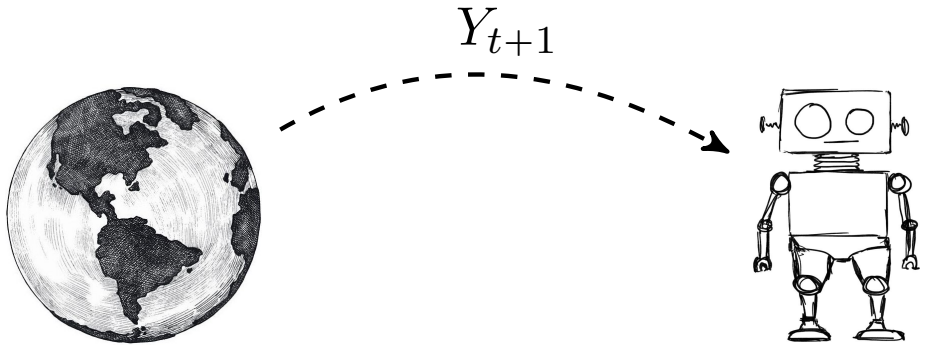
↳ *Again we assume a discrete set of possible rewards only for simplicity!*

Problem Formulation: *Reward*



+ *Does agent always get a reward?*
- No! It may have no reward in some time steps
  - ↳ *We could think of zero (empty) reward in that case, e.g.,* $R_{t+1} = 0$

Problem Formulation: *Observation*

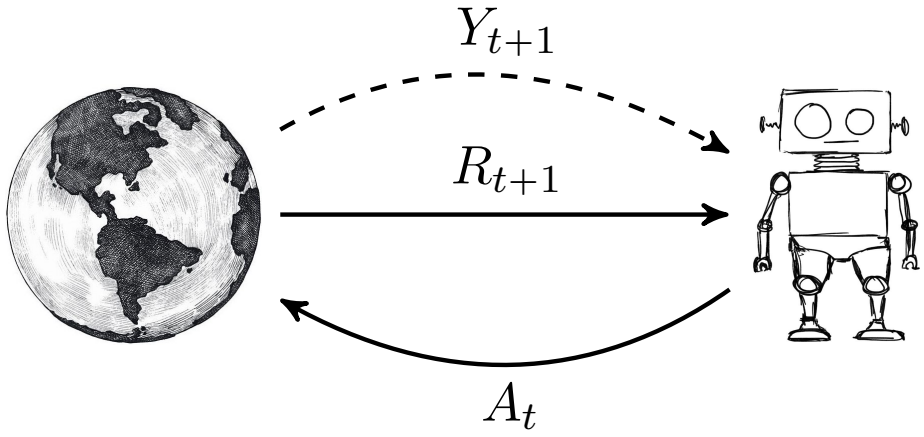


Before the next action, the agent also observes the environment

$$Y_{t+1} \in \mathbb{Y}$$

We index by $t+1$, as agent observes after each action

Problem Formulation: *Trajectory*



Let's make an agreement[1]

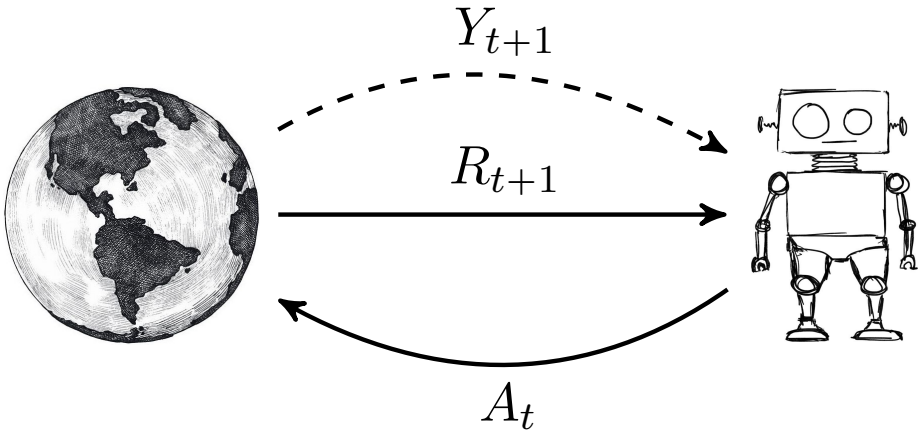
- *At time $t = 1$ agent observes initial observation Y_1 and reward R_1*
  - *It then decides for action A_1*
- *As a consequence it gets observation Y_2 and reward R_2*
  - *It then decides for action A_2*

...

This behavior is going to continue for ever, i.e., $t = 1, \ldots, \infty$

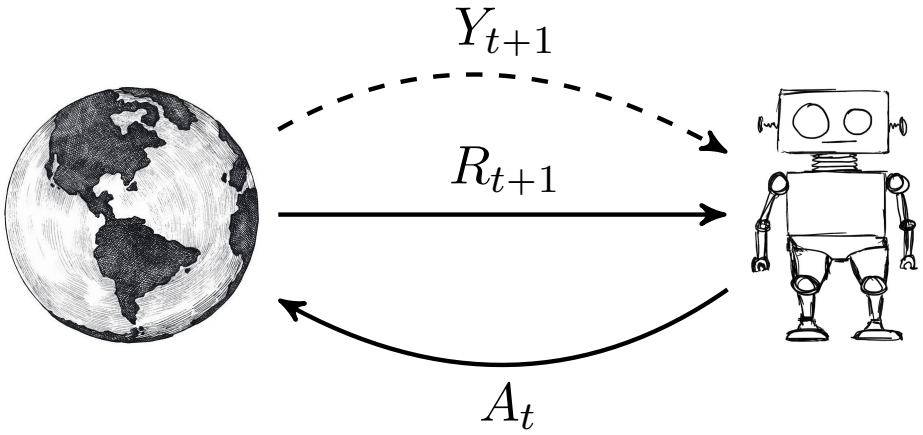
[1] We keep it as in *Sutton's book*, as it's a common notation

Problem Formulation: *Trajectory*



+ *But, shouldn't the agent stop at some point?!*
- Let's assume for now that *it never stops once started*
  - ↳ *We later define the concept of state*
  - ↳ *With this concept we can make agent stop without terminating t*
    - ↳ *The agent simply gets into a recurring state*

Problem Formulation: *History*

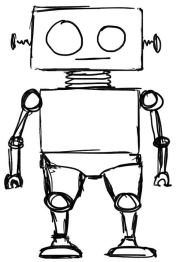


History contains all happened up to time t, before agent acts in time t

$$H_t = A_0, R_1, Y_1, A_1, R_2, Y_2, A_2, \ldots, R_t, Y_t$$

This is the information that has been collected by the agent up to time t

Problem Formulation: *Final Goal of Agent*



The final goal of the agent is to

maximize what it is going to collect as reward in future

+ *Why does agent looks at cumulative future reward?*
- Well it's a bit of philosophical argument: *we assume that*
  - ↳ *reward is all the agent would think about*
  - ↳ *the agent want to aggregate as much reward as possible*
  - ↳ *past has passed! Agents tries to maximize what it could get*

But, you have all the right *not to agree with this statement!*

↳ *In this case, you may look at it as a mathematical model* ☺

Problem Formulation: *Future Return*

We formulate *accumulated future reward* using the notion of return

Future Return

At time t, the future return of the agent is

$$G_t = \sum_{i=0}^{\infty} \gamma^i R_{t+i+1}$$

for some constant $0 \leqslant \gamma \leqslant 1$ referred to as discount factor

+ *What exactly is this return? Why do we have a discount factor?*
– Let's open it up!

Problem Formulation: *Future Return*

The future return at time t looks like

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

We can look at it this way: after we decide to act A_t at time t

- *We are going to get immediate reward R_{t+1} in next time step*
- *This will also impact our next action A_{t+1}*
  - ↳ *after this we will get R_{t+2}*
  - ↳ *after the next action we get R_{t+3}*

  ...
- *We rather prefer to get the reward sooner than later!*
  - ↳ *large R_{t+1} is preferred to large R_{t+2}*
  - ↳ *large R_{t+2} is preferred to large R_{t+3}*

  ...
- *We scale down gained reward with discount factor after each time step*

Problem Formulation: *Future Return*

The future return at time t looks like

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

Extreme choices of discount factor γ are zero and one:

- *Shortsighted agent sets $\gamma = 0$*
  - ↳ *It only cares about the immidiate reward*
- *Forethoughtful agent that sets $\gamma = 1$*
  - ↳ *It cares about the cumulative sum over time*
  - ↳ *It cares about future as much as it cares about now*
  - ↳ *It would accept less reward now, if it expects larger compensation in future*

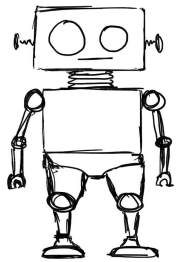
Problem Formulation: *Agent's Learning Task*

Given the concept of *future return*, we can formulate agent's goal as

> *At time t, agent looks at history H_t and decides for an action A_t that*
>
> *maximizes the future return G_t*

+ *But wait a moment! How can the agent optimize? How can it even know what the return value is?*
– We should model relations between the interaction means

A Stochastic Framework



Agent looks at the environment as a random object

- *Depending on agent's action, it returns a random reward and observation*
  - ↳ *Recall the multi-armed bandit example*
    - ↳ *Depending on choice of company, agent gets a random payment*

Agent may also act randomly

- *It is due to the randomness of environment*
  - ↳ *Recall the multi-armed bandit example*
    - ↳ *Agent tries each company for d days and decides after $2d$ days*
    - ↳ *It could go to Company A or B*
    - ↳ *The probability of going to Company A is higher though*

Modeling Components: *State*

Let's start modeling this stochastic framework: *we start with history*

> *History at time t is collection of what agent has observed up to that time*
>
> $$H_t = A_0, R_1, Y_1, A_1, R_2, Y_2, A_2, \ldots, R_t, Y_t$$

But this is a long sequence as $t \to \infty$

- *It cannot be stored into a finite memory!*
  - ↳ *For any memory-size, there is a time after which the memory is full*
- *Not all of these entries in the history are useful*
  - ↳ *Information context of the history is always limited*

We need to replace history with a finite memory component

+ *This reminds me of some relevant discussion in deep learning!*
– Yes! *State-dependent systems in recurrent neural networks*

Modeling Components: *State*

We assume that environment is a state-dependent system

State-dependent Environment

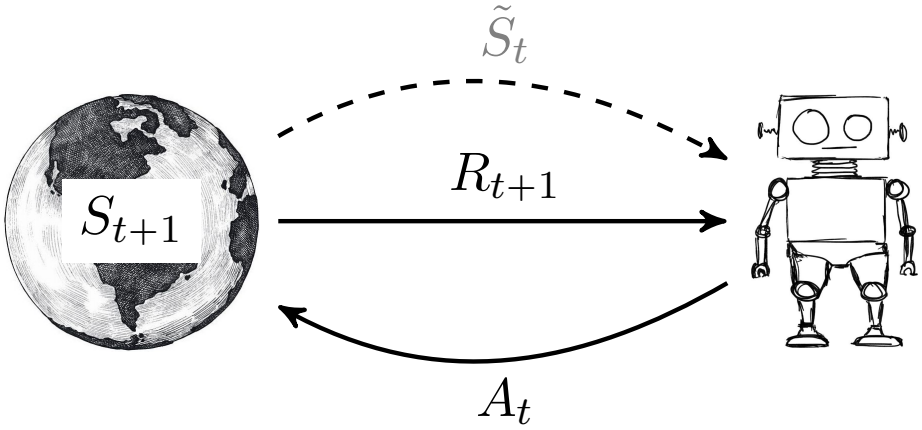
The environment at each time step has a particular state

$$S_t \in \mathbb{S} = \left\{s^1, \ldots, s^N\right\} \leftsquigarrow \text{set of possible states}$$

which along with the action A_t specifies its next state the reward it returns

- \+ *Is there necessary a discrete set of possible states?*
- – Not necessarily! *But we keep assuming whenever needed*
- \+ *How does this assumption impact the RL setting?!*
- – Let's see

RL Framework: *State-Representation*

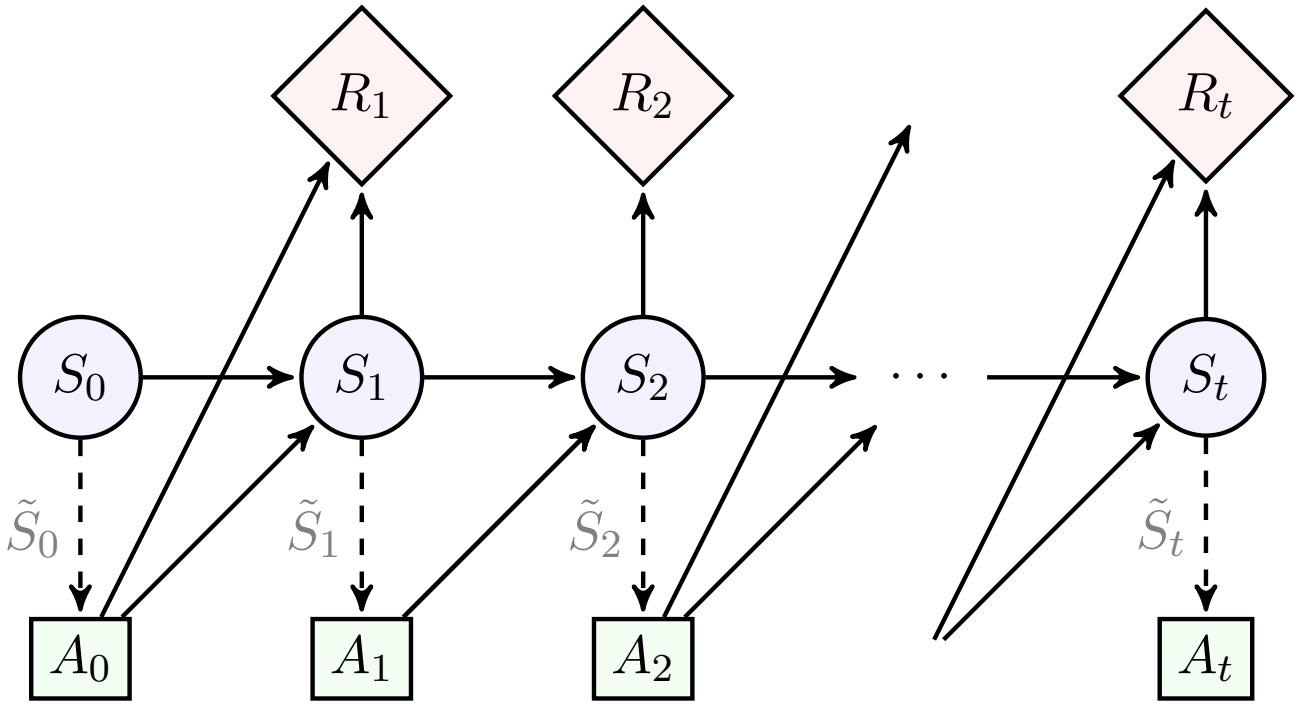


Let's say we are at time t and the environment is in state S_t

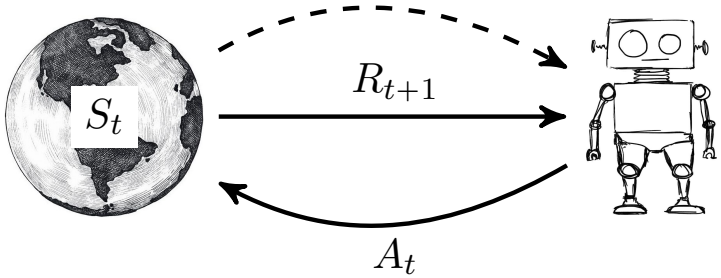
1. *Agent observes $\tilde{S}_t$*
   - ↳ *In ideal world, it can see the true state: $\tilde{S}_t = S_t$*
   - ↳ *In real world, it might see an incomplete version: $\tilde{S}_t \neq S_t$*
2. *Based in the observed state $\tilde{S}_t$, agents decides to act A_t*
3. *Environment receives action A_t*
   - ↳ *It rewards the agent as R_{t+1}*
   - ↳ *It changes its state to S_{t+1}*

RL Framework: *State-Representation*

So, we can plot the trajectory as a causal graph



RL Framework: *Markovity of State*



The *complete state* should *give all information about the environment at each time: this means that at time* t

the reward of next time R_{t+1} *and the next state* S_t

should be described completely by the action A_t *and current state* S_t

---

+ *Can we guarantee that we always find such state?*
– Well, we can assume that it exists, but maybe *we do not have access to it*

RL Framework: *Markovity of State*

We assume that the state S_t contains all information up to time t

↳ *This state describes a Markov process*

Markov Process

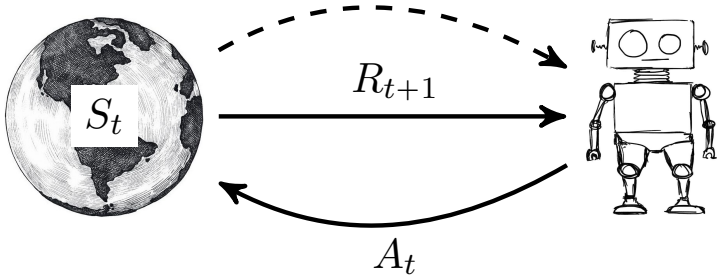
Sequence $S_1 \to S_2 \to \ldots$ describe a Markov process if

$$\Pr\{S_{t+1} = s_{t+1} | S_t = s_t, \ldots, S_1 = s_1\} = \Pr\{S_{t+1} = s_{t+1} | S_t = s_t\}$$

In other words: we assume that

whatever we need to know about past is always in the last state

RL Framework: *State-Representation*

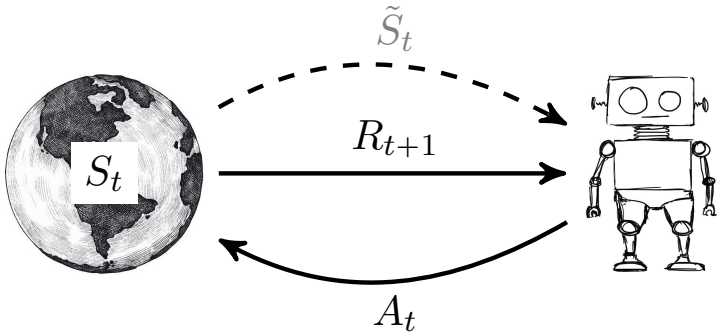


Moral of Story

Environment behaves based on a state which is a Markov process

- *At each time agent observes the state and acts based on that*
  - ↳ *It could be an incomplete observation*
- *Environment looks only on its current state and agent's action*
  - ↳ *it returns a reward*
  - ↳ *it changes its state*

RL Framework: *Agent's Observation*

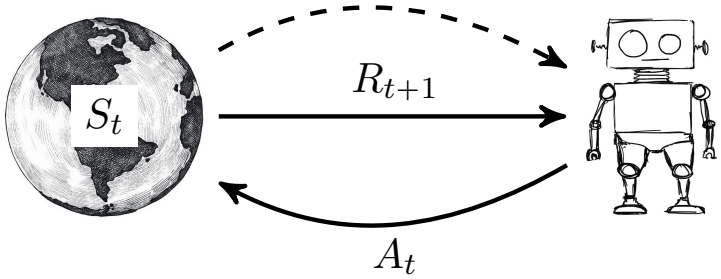


The agent could have different levels of access to the state

- *Full observation of the state:* $\tilde{S}_t = S_t$
  - ↳ *This is an optimal case, but does not happen always*
- *Partial observation of the state:* $\tilde{S}_t \neq S_t$
  - ↳ *This is more realistic, as some agents are restricted in their observation*
    - ↳ *A robot can only see a limited part of its surrounding area through its camera*

To keep things easy, we consider full observation for the moment

RL Framework: *Environment's Behavior*



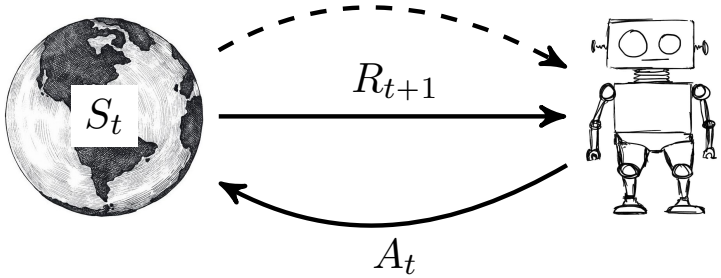
As we said: *environment decides for rewards and new state based on its current state and agent's action, we can describe this behavior mathematically by*

1. *Rewarding function that maps state S_t and action A_t to reward R_{t+1}*

$$\mathcal{R}(\cdot) : \mathbb{S} \times \mathbb{A} \mapsto \left\{r^1, \ldots, r^L\right\}$$

This mapping is in general random

RL Framework: *Environment's Behavior*



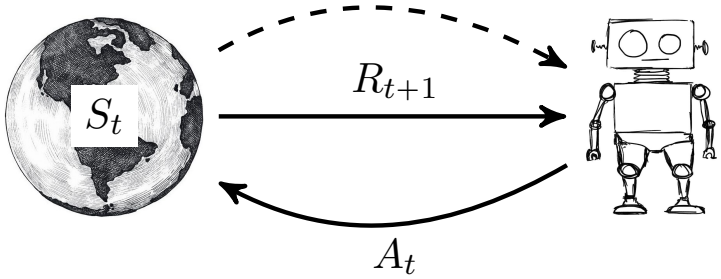
As we said: *environment decides for rewards and new state based on its current state and agent's action, we can describe this behavior mathematically by*

2. *Transition function* that maps *state* S_t and *action* A_t to the next state S_{t+1}

$$\mathcal{P}(\cdot) : \mathbb{S} \times \mathbb{A} \mapsto \mathbb{S}$$

This mapping is in general random

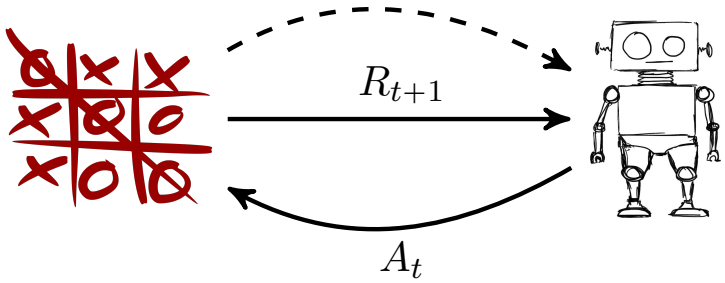
RL Framework: *Environment's Behavior*



+ *How do we get access to these mappings?*
- It depends on the problem
  - ↳ *In some problems, we can find these mappings based on the rules of the problem, e.g., board games or physical dynamic systems*
  - ↳ *In some others, we simply don't know! We may assume some mappings or try to solve the problem without using these mappings*

OK! Let's take a break and see some examples

Example: *Tic-Tac-Toe*

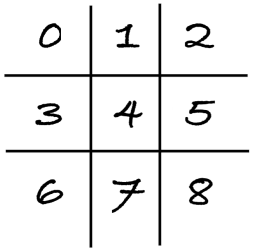


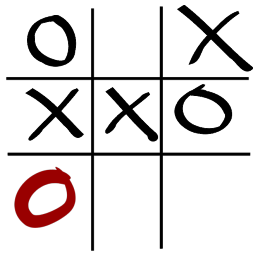
Let's consider the game of Tic-Tac-Toe: *in this game,*

- *Agent is one of the players*
  - ↳ *Action is the move done by the player*
- *Environment is the opponent and the game rules*
  - ↳ *State in each time is the status of the board*

Let's make it mathematically consistent!

Example: *Tic-Tac-Toe*





$S_t = \{0, 2, 3, 4, 5\}$

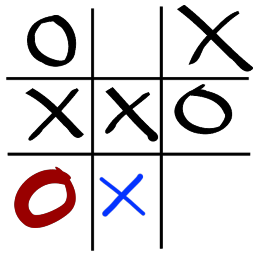
$A_t = 6$

We can number each cell on the board; in this case,

- *Played cells can be considered the sate*
  - ↳ *State in each time is the status of the board*
- *Next cell played is the action*

+ *Is it consistent with the Markovity assumption?*
- Sure! Let's see

Example: *Tic-Tac-Toe*



$S_t = \{0, 2, 3, 4, 5\}$
$A_t = 6$
$\rightsquigarrow S_{t+1} = \{0, 2, 3, 4, 5, 6, 7\}$

The player decides only based on the status of the board

↳ *Its action at time t relies on state S_t*

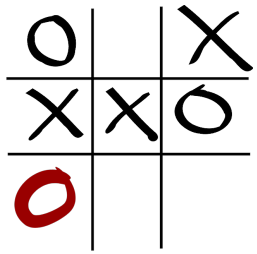
Next state also depends only on the current status and player's move

↳ S_{t+1} *is completely described by state S_t and action A_t*

$$S_{t+1} = S_t \cup \{A_t, O_{t+1}\}$$

with O_{t+1} being opponent's move

Example: *Tic-Tac-Toe*



$$S_t = \{0, 2, 3, 4, 5\}$$
$$A_t = 6$$

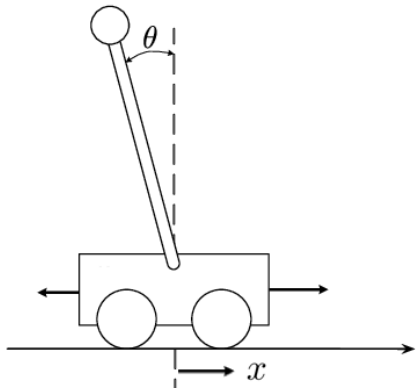
Transition function *is probabilistic in this case in this case*

$$\mathcal{P}\left(S_t, A_t\right) = \begin{cases} \{0, 2, 3, 4, 5, 6, 1\} & \textit{with probability } \Pr\{O_{t+1} = 1\} \\ \{0, 2, 3, 4, 5, 6, 7\} & \textit{with probability } \Pr\{O_{t+1} = 7\} \\ \{0, 2, 3, 4, 5, 6, 8\} & \textit{with probability } \Pr\{O_{t+1} = 8\} \end{cases}$$

Rewarding function *is deterministic*

$$\mathcal{R}\left(S_t, A_t\right) = \begin{cases} +1 & \textit{if player wins after playing } A_t \\ -1 & \textit{if player loses after playing } A_t \\ 0 & \textit{if players draw or the game is not over} \end{cases}$$

Example: *Cart-Pole Problem*



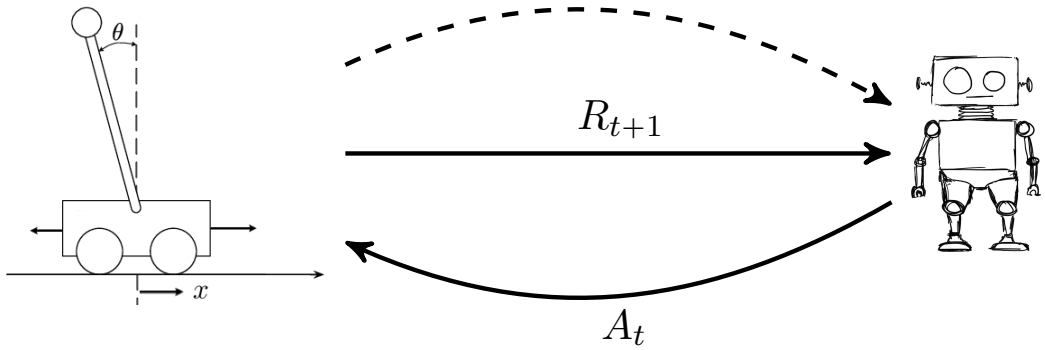
A pendulum is located vertically on a cart; at each time step, player can

- *either shift the cart to the right*
- *or shift it to the left*

The player gets \$1 for each time step that

- *the pendulum remains in an angle less than θ from the vertical line, and*
- *the cart is displaced with distance less than x*

Example: *Cart-Pole Problem*



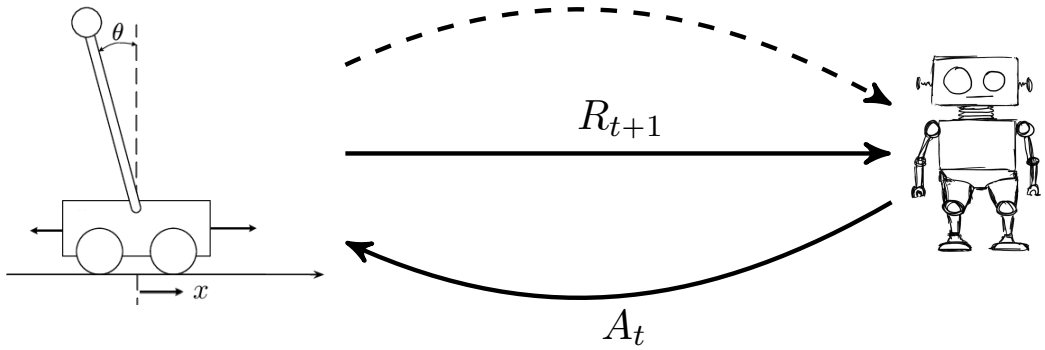
Agent is the player

- At time t, it acts as $A_t = \pm 1$, i.e., right or left

Environment is the *cart-pole, game rule and the physical laws*

- *State* at time t is the collection of *physical parameters*
  - ↳ Distance of cart, its velocity, angle of pendulum, angular velocity, . . .
  - ↳ *Current state* and *action completely* describe *next status* of pendulum
    - ↳ This is a *Markov state*
- It returns *reward* $R_{t+1} = 1$, if game is not over after *action* A_t

Example: *Cart-Pole Problem*



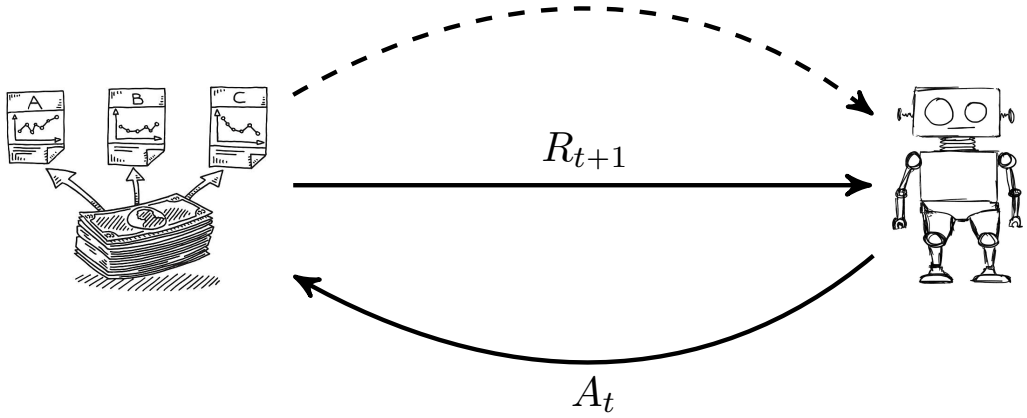
Transition function *is completely described by dynamics of the system*

$$\mathcal{P}\left(S_t, A_t\right) = \textit{Solution to dynamic equations}$$

Rewarding function *is further specified by the game rules*

$$\mathcal{R}\left(S_t, A_t\right) = \begin{cases} +1 & \textit{if game is not over after playing } A_t \\ 0 & \textit{otherwise} \end{cases}$$

Example: *Trading*



Agent is the one who invests

- *At time t, it acts by either buying or selling*

Environment is the market, investing portfolio, and transaction rules

- *State is a collection of market and portfolio features*
- *Reward R_{t+1} describes the profit made in each time frame*

Obviously in this case, transition function is not clear to us!

Playing in RL Framework: *Policy*

As environment gets to state S_t, agent decides for an action

↳ *We can describe this behavior via policy function*

Policy

Policy $\pi(a|s)$ *is a probability distribution over* a *conditional to* s

$$\pi(a|s) = \Pr\{A_t = a|S_t = s\}$$

+ *Why a probability distribution? Does agent always act randomly?!*
– It can either act randomly or deterministically, and *both these cases can be presented by this notation*

Playing in RL Framework: *Probabilistic Policy*

Assume we have set of possible actions

$$\mathbb{A} = \{a^1, \ldots, a^M\}$$

Agents might behave probabilistic, e.g.,

$$\pi(a|s) = \begin{cases} 0.8 & a = a^1 \\ 0.2 & a = a^2 \\ 0 & \text{otherwise} \end{cases}$$

This means that in state s

- *in 80% of cases it acts a^1*
- *in 20% of cases it acts a^2*
- *it never does any other action*

Playing in RL Framework: *Deterministic Policy*

Assume we have set of possible actions

$$\mathbb{A} = \{a^1, \ldots, a^M\}$$

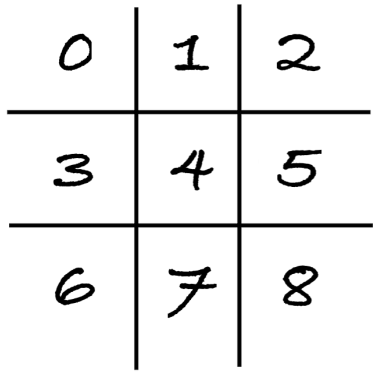
Agents might behave deterministically, e.g.,

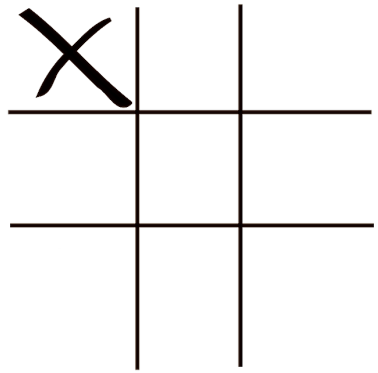
$$\pi(a|s) = \begin{cases} 1 & a = a^1 \\ 0 & \textit{otherwise} \end{cases}$$

This means that if the agent observes state s

- *it always acts a^1*
- *it never does any other action*

Example: *Tic-Tac-Toe*





Let's look at the Tic-Tac-Toe example: assume the oponent starts at cell 0

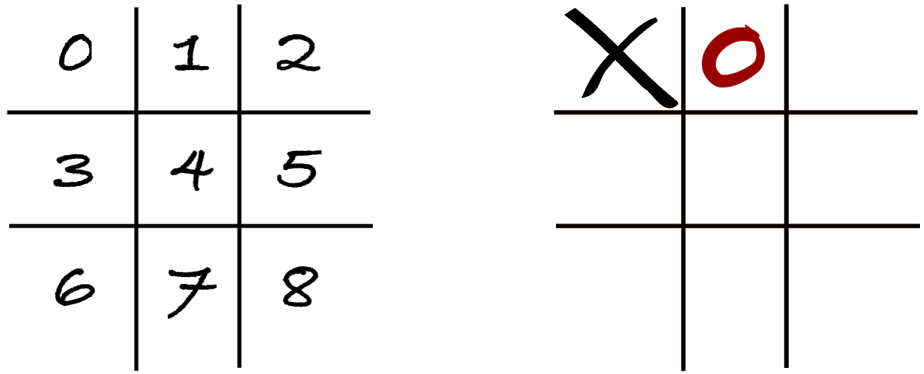
- *State is then* $S_1 = \{0\}$*: we can set the policy to*
  - ↳ *In the policy, we should have specified*

$$\pi\left(a|\left\{0\right\}\right) = \Pr\left\{A_1 = a | S_1 = \{0\}\right\} = \begin{cases} p_a & a = 1, \ldots, 8 \\ 0 & a = 0 \end{cases}$$

for some p_a*'s that satisfy*

$$\sum_{a=1}^{8} p_a = 1$$

Example: *Tic-Tac-Toe*

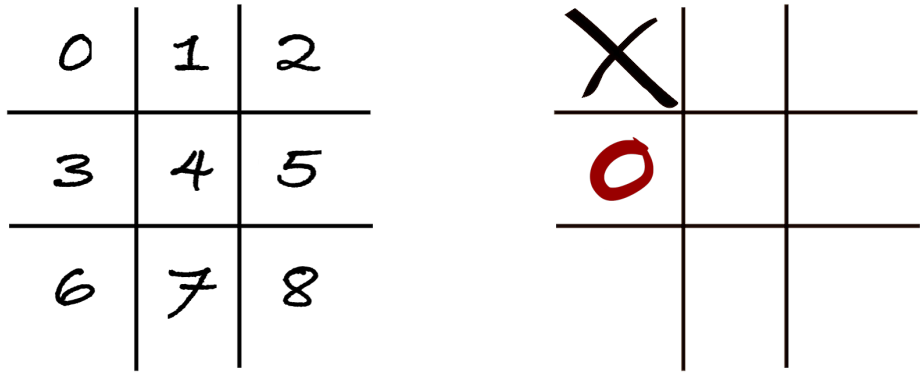


We may play deterministically: we are sure what to do after seeing opponent playing zero

↳ *We may always play* $A_1 = 1$

$$\pi\left(a|\left\{0\right\}\right) = \begin{cases} 1 & a = 1 \\ 0 & a \neq 1 \end{cases}$$

Example: *Tic-Tac-Toe*

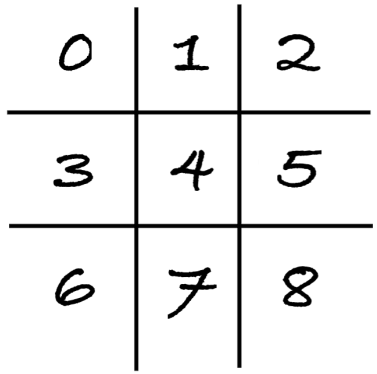


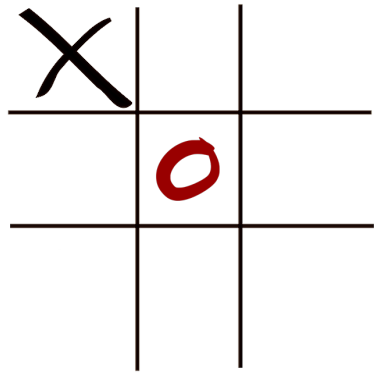
We may play deterministically: we are sure what to do after seeing opponent playing zero

↳ *We may always play* $A_1 = 3$

$$\pi\left(a|\left\{0\right\}\right) = \begin{cases} 1 & a = 3 \\ 0 & a \neq 3 \end{cases}$$

Example: *Tic-Tac-Toe*



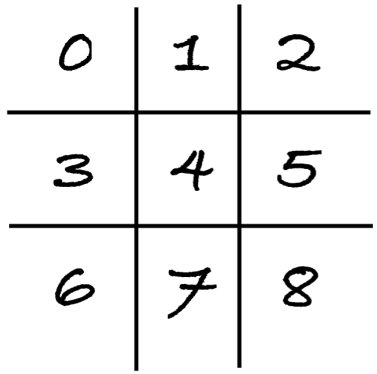


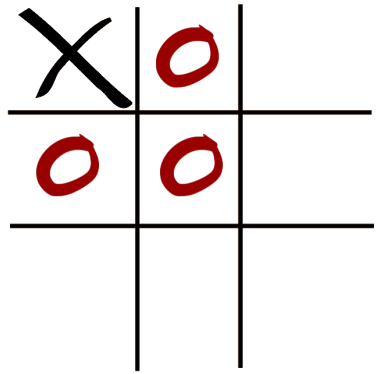
We may play deterministically: we are sure what to do after seeing opponent playing zero

↳ *We may always play* $A_1 = 4$

$$\pi\left(a|\left\{0\right\}\right) = \begin{cases} 1 & a = 4 \\ 0 & a \neq 4 \end{cases}$$

Example: *Tic-Tac-Toe*



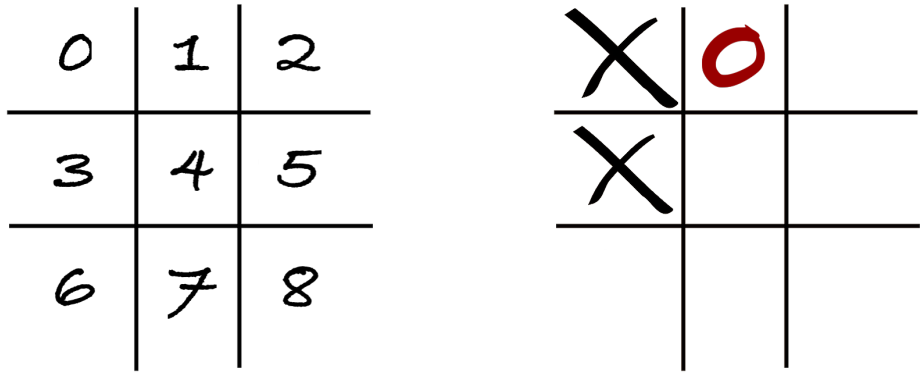


We may play probabilistic: we behave somehow randomly after seeing opponent playing zero

↳ *We may play any next cell* $A_1 \in \{1, 3, 4\}$

$$\pi\left(a \middle| \{0\}\right) = \begin{cases} 1/3 & a \in \{1, 3, 4\} \\ 0 & a \notin \{1, 3, 4\} \end{cases}$$

Example: *Tic-Tac-Toe*

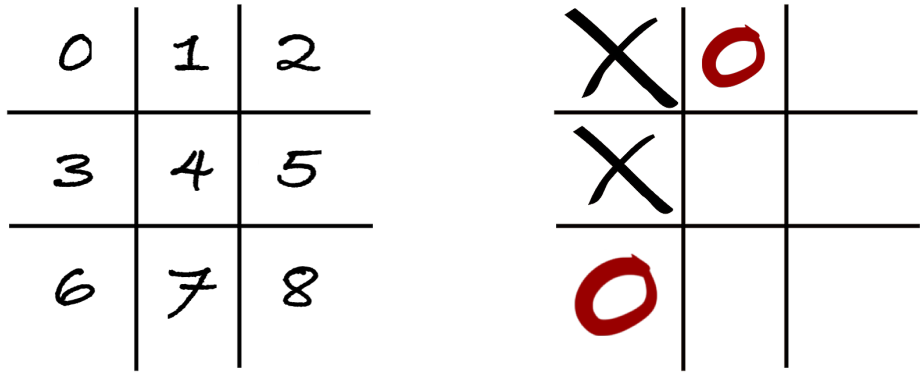


Say we acted $A_1 = 1$*, and the opponent plays* 3 *after our move: policy should tell us the next move*

↳ *State changes to* $S_2 = \{0, 1, 3\}$ *and our policy should say*

$$\pi\left(a|\left\{0,1,3\right\}\right) = \Pr\left\{A_2 = a | S_2 = \{0,1,3\}\right\}$$

Example: *Tic-Tac-Toe*



Say we acted $A_1 = 1$, and the opponent plays 3 after our move: policy should tell us the next move

↳ *For instance, we could play deterministically as*

$$\pi\left(a \middle| \{0, 1, 3\}\right) = \begin{cases} 1 & a = 6 \\ 0 & a \neq 6 \end{cases}$$

i.e., we always circle cell 6 in this state of the board

Playing in RL Framework: *Value Function*

As agent gets into a new state, it can use its policy to *estimate future return*

↳ *We call this estimate the value of the state*

Value Function

Assume that agent acts with policy $\pi(a|s)$ at state s; then, the value of this state is defined as

$$v_\pi(s) = \mathbb{E}\{G_t | S_t = s\}$$

Because it depends on policy π

+ *Why do we have index π?*
– Because this value depends on the policy!

Playing in RL Framework: *Value Function*

Let's break it down: *we first recall that*

Future return at time t

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

for some discount factor $0 \leqslant \gamma \leqslant 1$

Also, we recall that

Rewarding function is going to determine the next reward

$$R_{t+1} = \mathcal{R}\left(S_t, A_t\right)$$

and transition function determines the next state

$$S_{t+1} = \mathcal{P}\left(S_t, A_t\right)$$

Playing in RL Framework: *Value Function*

Value of state s is

$$v_\pi(s) = \mathbb{E}\{G_t | S_t = s\} = \mathbb{E}\left\{R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s\right\}$$

Say we are at time t: *at this time we see state* $S_t = s$

- *Policy* $\pi(a|s)$ *tells us what to play next*
  - ↳ *For instance, it deterministically specifies* A_t
- *Rewarding function specifies the reward of next time step*

$$R_{t+1} = \mathcal{R}(S_t, A_t)$$

- *Transision function specifies the next state*

$$S_{t+1} = \mathcal{P}(S_t, A_t)$$

  - ↳ *Both rewarding and transition could be probabilistic*

Playing in RL Framework: *Value Function*

Value of state s is

$$v_\pi(s) = \mathbb{E}\{G_t | S_t = s\} = \mathbb{E}\left\{R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s\right\}$$

Given rewarding function and policy: *distribution of R_{t+1} is found*

- *We compute the probability of each possible rewards, i.e.,*

$$p_\ell(s, a) = \Pr\left\{R_{t+1} = r^\ell | S_t = s, A_t = a\right\}$$

for $\ell = 1, \ldots, L$ and each possible action a

↳ *We can hence compute the first term*

$$\mathbb{E}_\pi\{R_{t+1} | S_t = s\} = \sum_{\ell=1}^{L} \sum_{m=1}^{M} p_\ell(s, a^m) \pi(a^m | s) r^\ell$$

Playing in RL Framework: *Value Function*

Note that *distribution of* R_{t+1} *can change, if we change policy*

Example: *Say rewarding function says* $\mathcal{R}\left(s, a^1\right) = 0$ *and* $\mathcal{R}\left(s, a^2\right) = 1$; *now, consider the following policies*

$$\pi^1(a|s) = \begin{cases} 1 & a = a^1 \\ 0 & a \neq a^1 \end{cases} \qquad \pi^2(a|s) = \begin{cases} 1 & a = a^2 \\ 0 & a \neq a^2 \end{cases}$$

Therefore, we could say

$$\mathbb{E}_{\pi^1}\{R_{t+1}|S_t = s\} = 0 \qquad \mathbb{E}_{\pi^2}\{R_{t+1}|S_t = s\} = 1$$

Playing in RL Framework: *Value Function*

Value of state s is

$$v_\pi(s) = \mathbb{E}\{G_t | S_t = s\} = \mathbb{E}\left\{R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s\right\}$$

For time $t+1$: *we have specified state S_{t+1} via transition function*

- *Policy $\pi(a|S_{t+1})$ tells us what to play next, i.e., A_{t+1}*
  - ↳ *Note that S_{t+1} is in general a random variable*
- *Rewarding function specifies the reward of next time step*

$$R_{t+2} = \mathcal{R}(S_{t+1}, A_{t+1})$$

↳ *We can hence compute the second term*

$$\mathbb{E}_\pi\{R_{t+2} | S_t = s\} \leftsquigarrow \text{Expectation over } S_{t+1}, A_{t+1} \text{ and } A_t$$

Playing in RL Framework: *Value Function*

Value of state s is

$$v_\pi(s) = \mathbb{E}\{G_t | S_t = s\} = \mathbb{E}\left\{R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s\right\}$$

So if we set a specific policy π, we could keep on

- $\mathbb{E}_\pi\{R_{t+3} | S_t = s\}$

  $\dots$

and we could estimate future return for a given policy π as

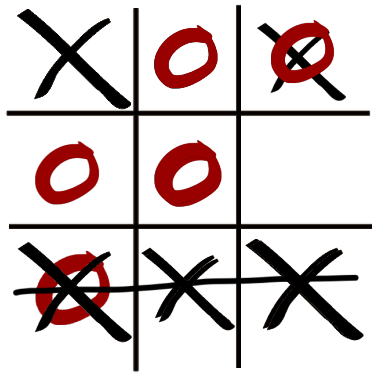
$$v_\pi(s) = \mathbb{E}_\pi\{R_{t+1} | S_t = s\} + \gamma \mathbb{E}_\pi\{R_{t+2} | S_t = s\} + \gamma^2 \mathbb{E}_\pi\{R_{t+3} | S_t = s\} + \cdots$$

Value Example: *Tic-Tac-Toe*

Assume we play Tic-Tac-Tao with a pretty deterministic opponent:

We know that opponent takes the following strategy:

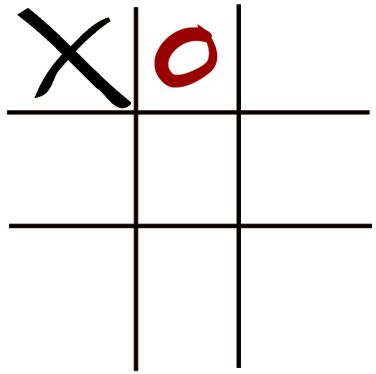
- *if corners are available without chance of loosing, it plays a corner*
  - ↳ *it starts with cross-corner*
  - ↳ *it plays other corners afterwards*
- *if a line can be filled by us in next move, it blocks the line*
  - ↳ *if multiple lines are available, it blocks one of them at random*
- *if there is a line to be filled, it fills and wins*



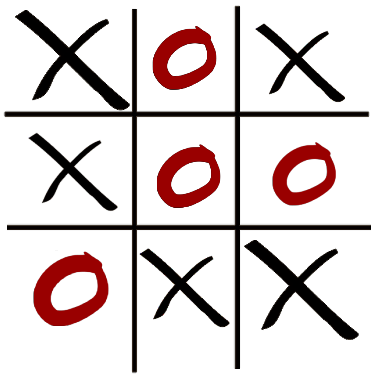
Value Example: *Tic-Tac-Toe – Policy I*

We now take the following policy

- *if there is no chance of loosing, we play next cell*
  - ↳ *if next cell in the row available, we play that cell*
  - ↳ *if next cell in the row not available, we play cell below*
  - ↳ *if neither is available, we play some cell at random*
- *if there is a line possibly filled in next move by opponent, we block it*
  - ↳ *if multiple lines are available, we block one of them at random*
- *if there is a line to be filled by us, we fill and win*



Value Example: *Tic-Tac-Toe – Policy I*



$v_{\pi^1}(\{0\}) = 0$

assuming $\gamma = 1$

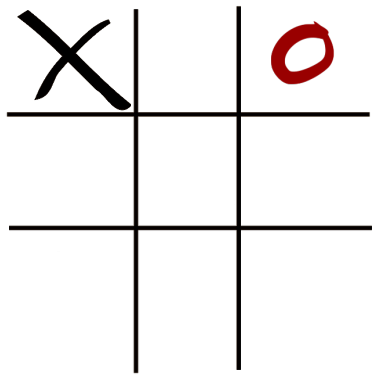
Say the game starts at state $s = \{0\}$ *with policy I:* π^1

- *Our next move will be* $A_0 = 1$
- *Opponent will play* $O_1 = 8$
  - ↳ *this leads to new state* $S_1 = \{0, 1, 8\}$
- *Our next move will be* $A_1 = 4$
- *Opponent will play* $O_2 = 7$
  - ↳ *this leads to new state* $S_2 = \{0, 1, 4, 7, 8\}$
- *Our next move will be* $A_2 = 6$
- *Opponent will play* $O_3 = 2$
  - ↳ *this leads to new state* $S_3 = \{0, 1, 2, 4, 6, 7, 8\}$
- *Our next move will be* $A_3 = 5$
- *Opponent will play* $O_4 = 3$
  - ↳ *this leads to new state* $S_4 = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$

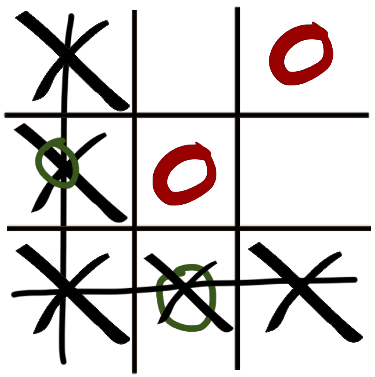
Value Example: *Tic-Tac-Toe – Policy II*

Let's now change the policy

- *if there is no chance of loosing, we play a corner*
  - ↳ *if a corner is available in the row, we play that corner*
  - ↳ *if no corner is available in the row, we play corner in the same column*
  - ↳ *if neither is available, we play some random cell*
- *if there is a line possibly be filled in next round by opponent, we block it*
  - ↳ *if multiple lines are available, we block one of them at random*
- *if there is a line to be filled by us, we fill and win*



Value Example: *Tic-Tac-Toe – Policy II*



$v_{\pi^2}(\{0\}) = -1$

assuming $\gamma = 1$

Say the game starts at state $s = \{0\}$ with policy I: π^2

- *Our next move will be $A_0 = 2$*
- *Opponent will play $O_1 = 8$*
  - ↳ *this leads to new state $S_1 = \{0, 2, 8\}$*
- *Our next move will be $A_1 = 4$*
- *Opponent will play $O_2 = 6$*
  - ↳ *this leads to new state $S_2 = \{0, 2, 4, 6, 8\}$*
- *Our next move will be random*
  - ↳ *with 50% chance we play $A_2 = 7$*
    - ↳ *Opponent will play $O_3 = 3$*
  - ↳ *with 50% chance we play $A_2 = 3$*
    - ↳ *Opponent will play $O_3 = 7$*
- *With any possible move, we end up losing*

Playing in RL Framework: *Value Function*

Value function can be also calculate for each possible next action

↳ *We call this function the action-value of the state-action pair*

Action-Value Function

Assume that agent acts with policy $\pi(a|s)$ at state s; then, the action-value of this state is defined as

$$q_\pi(s, a) = \mathbb{E}_\pi\{G_t | S_t = s, A_t = a\}$$

Because it depends on policy π

Playing in RL Framework: *Action-Value Function*

Action-Value of state-action pair (s, a) is

$$q_\pi(s, a) = \mathbb{E}_\pi \left\{ R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s, A_t = a \right\}$$

Given rewarding function: *distribution of R_{t+1} for given pair (s, a) is*

- *We compute the probability of each possible rewards, i.e.,*

$$p_\ell(s, a) = \Pr \left\{ R_{t+1} = r^\ell | S_t = s, A_t = a \right\}$$

for $\ell = 1, \ldots, L$ and each possible action a

↳ *We can hence compute the first term*

$$\mathbb{E}_\pi \left\{ R_{t+1} | S_t = s, A_t = a \right\} = \sum_{\ell=1}^{L} p_\ell(s, a) \, r^\ell$$

Playing in RL Framework: *Action-Value Function*

Action-Value of state-action pair (s, a) *is*

$$q_\pi(s, a) = \mathbb{E}_\pi \left\{ R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s, A_t = a \right\}$$

For time $t+1$: *we have specified state* S_{t+1} *via transition function*

- *Policy* $\pi(a|S_{t+1})$ *tells us what to play next, i.e.,* A_{t+1}
  - ↳ *Note that* S_{t+1} *is in general a random variable*
- *Rewarding function specifies the reward of next time step*

$$R_{t+2} = \mathcal{R}(S_{t+1}, A_{t+1})$$

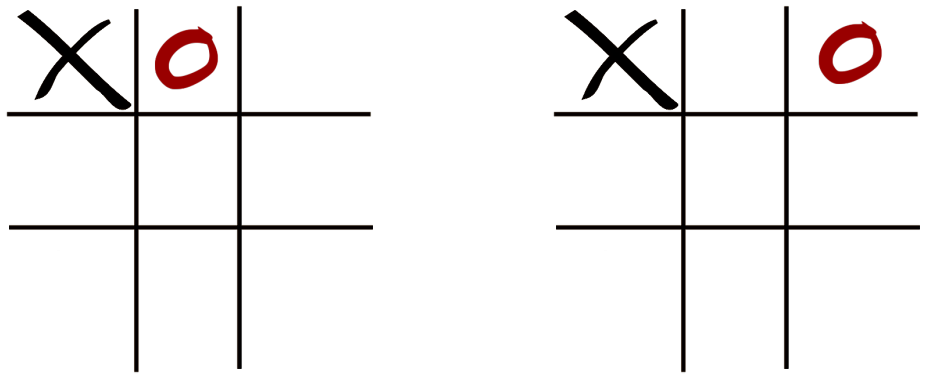
  - ↳ *We can hence compute the second term*

$$\mathbb{E}_\pi \{ R_{t+2} | S_t = s, A_t = a \} \leftsquigarrow \text{Expectation over } S_{t+1} \text{ and } A_{t+1}$$

Action-Value Example: *Tic-Tac-Toe – Hybrid Policy*

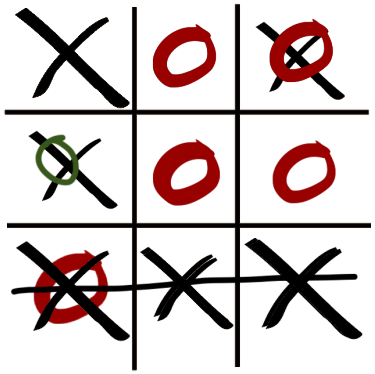
We play Tic-Tac-Tao with the same opponent: this time however we take a hybrid approach

- *with* 50% *chance we stick to policy I*
- *with* 50% *chance we stick to policy II*



Let's denote this hybrid policy by $\bar{\pi}$

Action-Value Example: *Tic-Tac-Toe – Hybrid Policy*



Say the game starts at state $s = \{0\}$

- *If we decide to stick to policy I*
  - ↳ *next move will be* $A_0 = 1$
  - ↳ *we end up drawing*

$$q_{\bar{\pi}}\left(\{0\}, 1\right) = 0$$

- *If we decide to stick to policy II*
  - ↳ *next move will be* $A_0 = 2$
  - ↳ *we end up losing*

$$q_{\bar{\pi}}\left(\{0\}, 2\right) = -1$$

We see it in the assignment that with this hybrid approach we have

$$v_{\bar{\pi}}\left(\{0\}\right) = 0.5 q_{\bar{\pi}}\left(\{0\}, 1\right) + 0.5 q_{\bar{\pi}}\left(\{0\}, 2\right) = -0.5$$

Playing in RL Framework: *Optimal Policy*

+ *What is the application of value functions?*
- They let us compare policies

We can compare π^1 to π^2 using the value function

Better Policy

We say policy π^1 is better than policy π^2, and denote it by $\pi^1 \geqslant \pi^2$ if

$$v_{\pi^1}(s) \geqslant v_{\pi^2}(s)$$

for all $s \in \mathbb{S}$

Playing in RL Framework: *Optimal Policy*

It is important to note that when we say $\pi^1 \geqslant \pi^2$; it means that by playing π^1, we expect higher or equal future return than π^2, no matter at which state we start using policy π^1

This definition helps us defining optimal policy: *simply speaking*

optimal policy is the policy that is the best

Optimal Policy

Policy $\pi^\star$ is called optimal policy if for any possible policy π, we have

$$\pi^\star \geqslant \pi$$

Playing in RL Framework: *Optimal Policy*

Optimal Policy: *Alternative Definition*

Policy $\pi^\star$ is called optimal policy if it maximizes the value for each state

$$\pi^\star = \underset{\pi}{\operatorname{argmax}}\, v_\pi\left(s\right)$$

for all $s \in \mathbb{S}$

+ *Can we guarantee that the optimal policy always exists?*
– Yes!
+ *Why didn't we define it based on action-value function?*
– Well! We could, but we end up with the same result!

Playing in RL Framework: *Optimal Policy*

Basic Property of Optimal Policy

Let $\pi^\star$ be the optimal policy; then,

$$v_\star(s) = v_{\pi^\star}(s) = \max_\pi v_\pi(s)$$

for all $s \in \mathbb{S}$, and

$$q_\star(s,a) = q_{\pi^\star}(s,a) = \max_\pi q_\pi(s,a)$$

for all $s \in \mathbb{S}$ and $a \in \mathbb{A}$

Attention

Optimal policy is not necessarily unique: we may have multiple optimal policies

Ultimate Goal in RL: *Finding Optimal Policy*

Ultimate goal in an RL problem is to find the optimal policy

But there are a bunch of challenges in this way

1. *How can we get a model of environment?*
   - ↳ *Maybe, we could assume some model*
     - ↳ *We assume some transition and rewarding functions*
     - ↳ *Our RL framework is based on a model*
   - ↳ *Maybe, we could estimate what we need directly from enough samples*
     - ↳ *We directly compute value or policy from samples, e.g.,*

$$q_\pi(s, a) = \mathbb{E}\{G_t | S_t = s, A_t = a\} \approx \text{average from samples}$$

     - ↳ *Our RL framework is free of a model*

High Level Classification of RL Methods

Model-Based RL

- Bellman Equation
- value iteration
- policy iteration

Model-free RL

- on-policy methods
- temporal difference
- Monte Carlo
- SARSA

- off-policy methods
- Q-learning

Ultimate Goal in RL: *Finding Optimal Policy*

Ultimate goal in an RL problem is to find the optimal policy

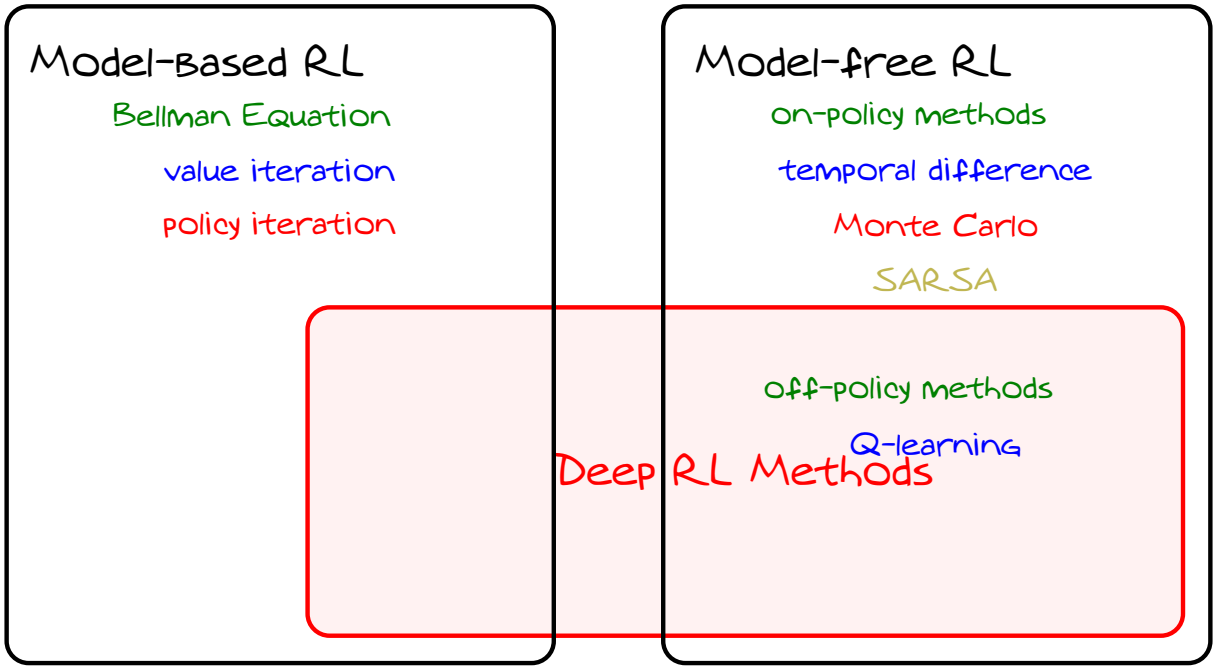
But there are a bunch of challenges in this way

2. *How can we apply either approach in environments with huge # of states?*
   - ↳ *Maybe, we could use neural networks*
     - ↳ *We approximate the target function via a neural network, e.g.,*

$$v_\star(s) \approx \mathrm{CNN}(s|\mathbf{w})$$

     - ↳ *We collect some samples and train the network based on them*
     - ↳ *We use that approximation instead of true function*

High Level Classification of RL Methods



Frozen Lake Game

The guy needs to get to the treasure

- *It could walk over frozen cells*
- *If it runs to a shallow cell it fells into water*

If he decides for a direction, it could

- *move to direction with probability* $1 - p$
- *slip with probability* p *to another direction*

It wants to learn the best way, i.e.,

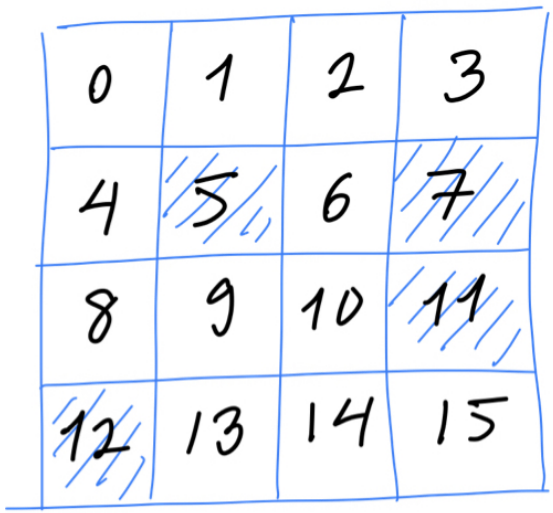
with minimal chance of failing

Playing in RL Framework

For this problem, we should first define *the main three components*

- *State*
- *Action*
- *Reward*

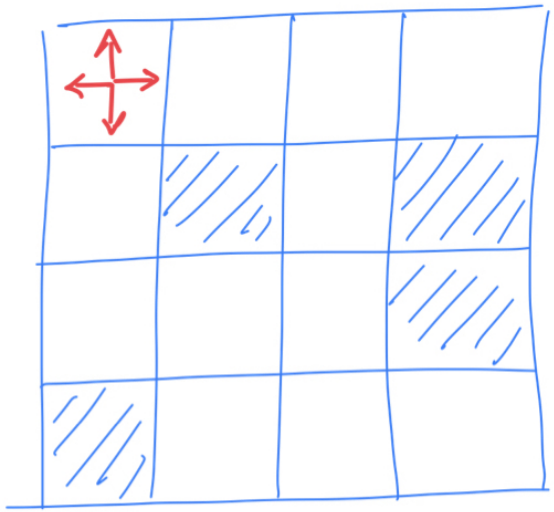
State $\equiv$ *Location of the guy on the game board*



$$\$ = \{0, 1, \ldots, 15\}$$

Playing in RL Framework

Action $\equiv$ *Possible moves on the board*

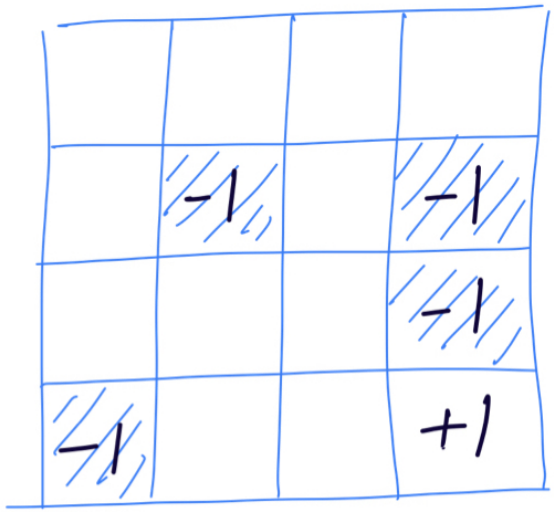


We can represent them with some integer

$$\mathbb{A} = \{0 \equiv \texttt{left}, 1 \equiv \texttt{down}, 2 \equiv \texttt{right}, 3 \equiv \texttt{up}\}$$

Playing in RL Framework

Reward $\equiv$ *Possible outcomes of the game*



We have three possible cases

$$R \in \{0 \equiv \text{ongoing}, 1 \equiv \text{winning}, -1 \equiv \text{losing}\}$$

Playing in RL Framework

+ *Can we describe the environment in this problem?*
– Sure!

Rewarding Function

Reward of each state-action pair is a random variable

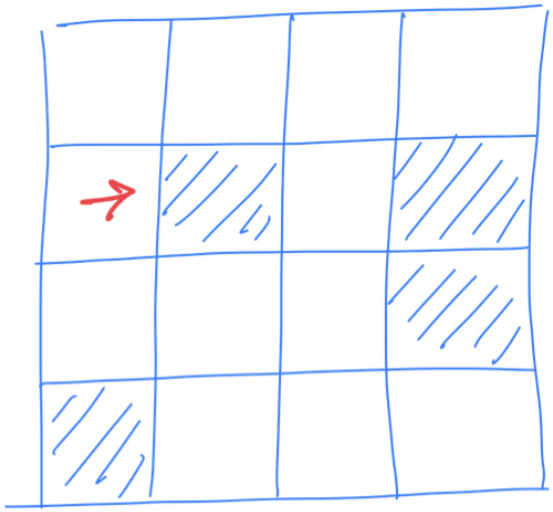
$$\mathcal{R}(s,a) = \begin{cases} \text{intended reward} & 1-p \\ \text{reward of slipping direction} & p \end{cases}$$

Transition Function

Transition of each state-action pair is also a random variable

$$\mathcal{P}(s,a) = \begin{cases} \text{intended state} & 1-p \\ \text{state of slipping direction} & p \end{cases}$$

Playing in RL Framework

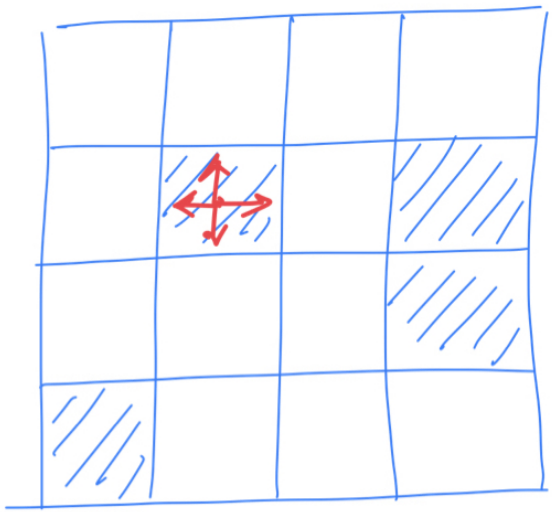


For example, in the above image we could say

$$\mathcal{P}(4,2) = \begin{cases} 5 & 1-p \\ 0 \text{ or } 8 & p \end{cases} \quad \textit{and} \quad \mathcal{R}(4,2) = \begin{cases} -1 & 1-p \\ 0 & p \end{cases}$$

Terminal State

+ *But, how can we specify the end of the game?*
– We need to define a terminal state



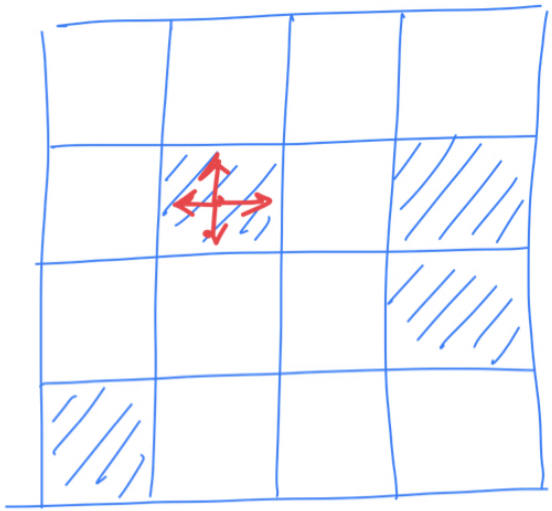
We can think of the end as a state which does not change anymore

$$\mathcal{P}(5, a) = 5$$

for all actions $a \in \mathbb{A}$

Terminal State

- \+ *What will be the reward then?*
- – We keep getting zero reward



We can think of the end as a state which does not change anymore

$$\mathcal{R}(5, a) = 0$$

for all actions $a \in \mathbb{A}$

Terminal State

Terminal State

State s is called terminal if for any action $a \in \mathbb{A}$, we have

$$\mathcal{P}(s, a) = s \qquad \text{and} \qquad \mathcal{R}(s, a) = 0$$

A basic property of terminal state is that *it has zero value*: *say s is terminal*

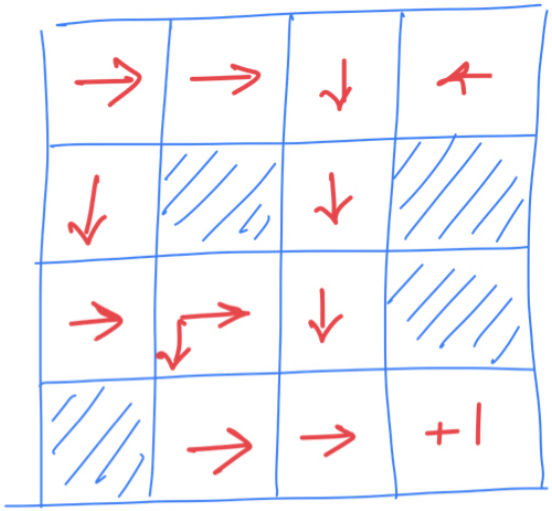
$$\begin{aligned} v_\pi(s) = \mathbb{E}_\pi\{G_t | S_t = s\} &= \mathbb{E}_\pi\left\{R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s\right\} \\ &= \mathbb{E}_\pi\left\{0 + \gamma 0 + \gamma^2 0 + \ldots | S_t = s\right\} = 0 \end{aligned}$$

Episode

The trajectory from starting state to a terminal state is often called an episode

Playing in RL Framework: *Policy*

Policy in this problem describes the *planned path towards the goal*

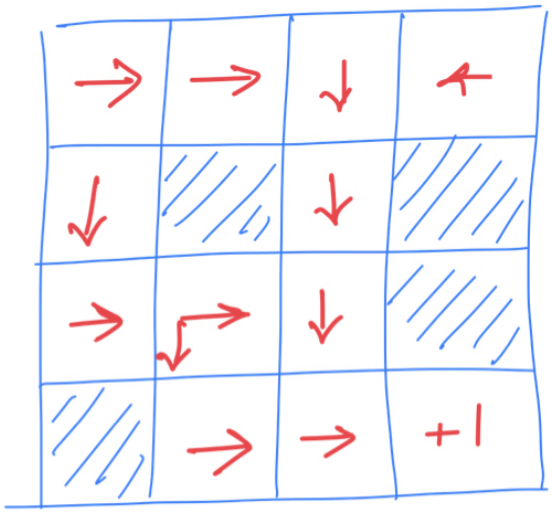


For instance in the above policy we have

$$\pi\left(a|1\right)=\begin{cases}1 & a=2\\ 0 & a\neq 2\end{cases} \quad \text{and} \quad \pi\left(a|9\right)=\begin{cases}0.5 & a=2\\ 0.5 & a=1\\ 0 & a=0,3\end{cases}$$

Playing in RL Framework: *Value*

Value describes the *chance of getting to the goal if we play the given policy*

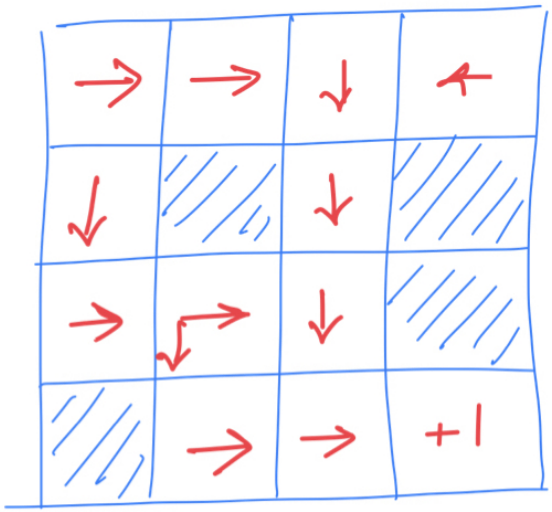


For instance, if slipping probability is $p = 0$ the above policy leads to

$$v_\pi(s) = \begin{cases} 1 & s \text{ is not terminal} \\ 0 & s \text{ is terminal} \end{cases}$$

Playing in RL Framework: *Optimal Policy*

Policy that maximizes the *chance of getting to the goal*



For instance, if slipping probability is $p = 0$ the above policy is optimal

Gymnasium

Gymnasium is an API with some pre-implemented environments

- *We can call environments and access the state and reward*
- *We can interact through actions*
- *This can is a powerful toolkit to test our RL algorithms*
  - ↳ *At some point, we may prefer implementing our own environment*

Gymnasium: *Trying Frozen Lake Game*

We can call the frozen lake environment easily in Gymnasium

```python
import gymnasium as gym

def test():
    # Create FrozenLake instance
    env = gym.make('FrozenLake-v1', map_name="4x4", is_slippery=False, render_mode='human')

    env.reset()  # Initialize to state 0
    terminated = False      # True when agent falls in hole or reached goal
    truncated = False       # True when agent takes more than 200 actions

    while(not terminated and not truncated):

        action = env.action_space.sample()

        # Execute action
        state, reward, terminated, truncated, _ = env.step(action)
        print(reward)
        print(state)
    env.close()
```

We will play around with it in Assignment 1