ÖZYEĞİN UNIVERSITY

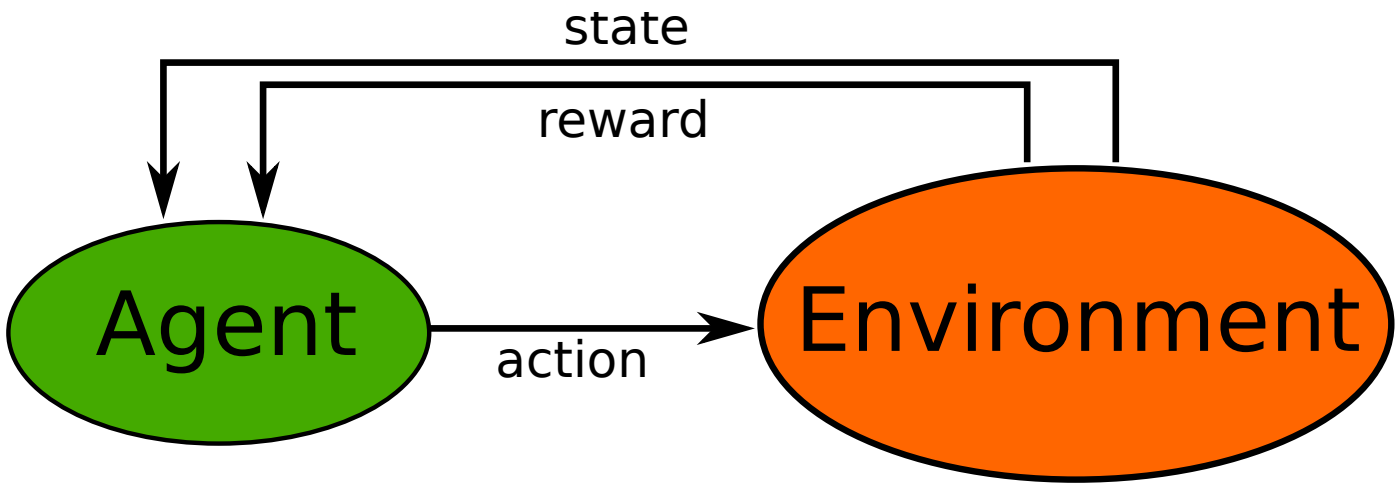
3 Markov Decision Processes

Melih Kandemir

Özyeğin University
Computer Science Department
melih.kandemir@ozyegin.edu.tr

10 Oct 2017

RL setup recap



Reinforcement learning is to

- learn from interaction to achieve a goal.
- in order words, update a policy wrt experience.

The boundary between the agent and the environment

- **Agent:** Anything that the system has power to change arbitrarily.
  (e.g. valid moves in chess)
- **Environment:** Anything on which the agent has limited or no control.
  (e.g. joint angles of a robot, engine horsepower in autonomous driving)

The precise boundary between the agent and environment varies in applications. Limbs of a robot are:

- the agent if we care only about high-level planning.
- the environment if we are to control them with RL.

The environment in math: MDPs

The Markov Decision Process (MDP) is

- a probabilistic model to describe the fully-observable environment in a format eligible to performing RL.
- effectively, 50 % of the course material is to understand basic properties of MDPs.

Rewards

- The primary concern of RL is to design goals by rewards (i.e. to describe what the goal is, not how to achieve it).
- Subgoals should not be rewarded at all if possible, should at least be rewarded much less than the main goal. The system might otherwise learn to satisfy with achieving only the subgoals.

Episodic and continuous tasks

- State sequences of **episodic tasks** break naturally (i.e. a chess game):

$$S_1, A_1, R_2, S_2, A_2, R_3, S_3, A_3, R_4, S_4 = s_e \text{ (Episode 1)}$$
$$S_1, A_1, R_2, S_2, A_2, R_3, S_3, A_3, R_4, S_4, A_4, R_5, S_5 = s_e \text{ (Episode 2)}$$
$$\dots$$
$$S_1, A_1, R_2, S_2, A_2, R_3, S_3 = s_e \text{ (Episode M)}$$

  where s_e is the end state.
- **Continuous tasks** never end.

The Markov Property

Given a *good enough description* of present, the future is independent of the past:

$$P(S_{t+1}|S_t) = P(S_{t+1}|S_1, \cdots, S_t).$$

- Does not mean we do not care about the past.
- Means we can encapsulate it in *some* careful definition of state.
- Encourages effective state design.
- Greatly simplifies the system of random variables we need to tackle.

The Markov Process

A tuple of two entities $\langle \mathcal{S}, \mathcal{P} \rangle$, where

- $\mathcal{S}$ is the set of environment states.
- $\mathcal{P} = P(S_{t+1}|S_t)$ is the **environment dynamics model**.
  Also known as the *transition probability distribution*.
  I will call it the *transition model*.

The Markov Reward Process

A tuple of four entities $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where

- $\mathcal{S}$ is the set of environment states: $S_t = s$ with $s \in \mathcal{S},\ \forall t$.
- $\mathcal{R}$ is the set of rewards: $R_t = r$ with $r \in \mathcal{R},\ \forall r$.
- $\gamma \in [0, 1]$ is the **discount factor**.
- $\mathcal{P} = P(R_{t+1}, S_{t+1}|S_t)$ is the **environment dynamics model** that naturally decomposes according to the chain rule as

$$P(R_{t+1}, S_{t+1}|S_t) = \underbrace{P(R_{t+1}|S_{t+1}, S_t)}_{Reward\ model} \times \underbrace{P(S_{t+1}|S_t)}_{transition\ model} .$$

We keep the assumption that the state transitions follow the Markov property

$$P(S_{t+1}|S_t) = P(S_{t+1}|S_1, \cdots, S_t).$$

Random variables of an episode

Take the episode below:

$$S_1, R_2, S_2, R_3, S_3$$

How would its joint distribution decompose?

Random variables of an episode

First follow the chain rule (this time in probability theory, not calculus):

$$
\begin{aligned}
&P(S_1, R_2, S_2, R_3, S_3) \\
&\quad = P(S_3, R_3, S_2, R_2 | S_1) P(S_1) \\
&\quad = P(S_3, R_3, S_2 | R_2, S_1) P(R_2 | S_1) P(S_1) \\
&\quad = P(S_3, R_3 | S_2, R_2, S_1) P(S_2 | R_2, S_1) P(R_2 | S_1) P(S_1) \\
&\quad = \underbrace{P(S_3 | R_3, S_2, R_2, S_1)}_{P(S_3|S_2)} \underbrace{P(R_3 | S_2, R_2, S_1)}_{P(R_3|S_2)} \underbrace{P(S_2 | R_2, S_1)}_{P(S_2|S_1)} \\
&\qquad P(R_2 | S_1) P(S_1). \\
&\quad = P(S_3 | S_2) P(R_3 | S_2) P(S_2 | S_1) P(R_2 | S_1) P(S_1).
\end{aligned}
$$

Red: Markov property.

Blue: Reward model.

Return

Cumulative discounted reward starting from timestep t on

$$G_t \triangleq R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}.$$

- $\gamma^k R_{t+k+1}$ is the *present* value of the *future* reward R_{t+k+1}.
- $\gamma = 0$: Myopic return model
- $\gamma = 1$: Far-sighted return model

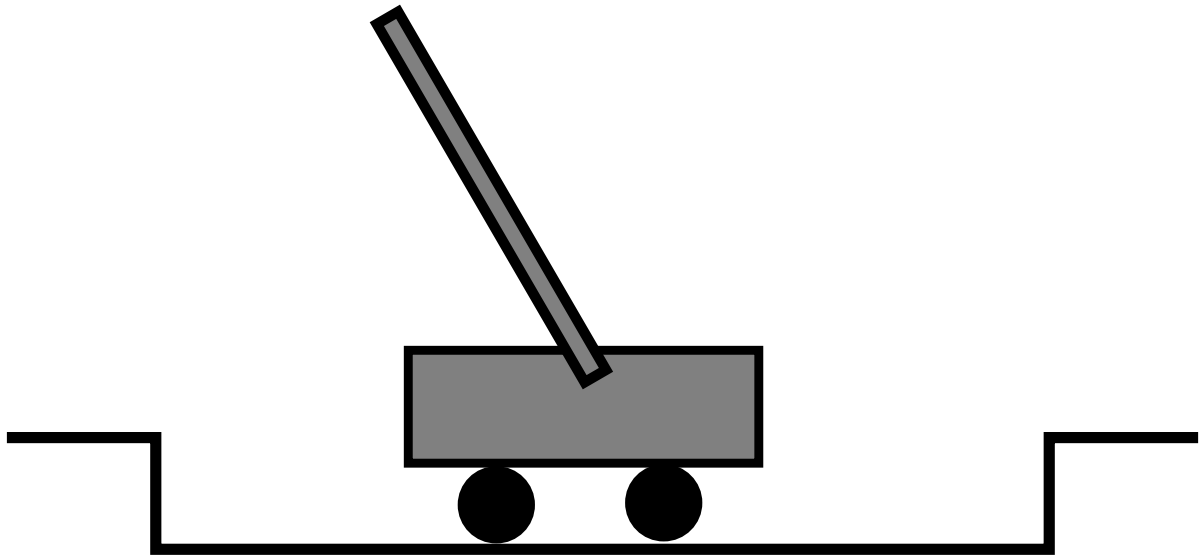
Why should we discount reward?

Account for two main risk factors:

- The environment model is not perfect.
  - Near future can be predicted more accurately than far future.
  - Make decisions based more on certain knowledge and less on uncertain knowledge (but based still on both).
- The learning model is not perfect.
  - We are up to *learning to decide*. The model itself will remain largely imperfect along the way.
  - How much would you rely on the advice of a child about how to invest your money for profit to come 20 years later? (what if it were an adult?)

Example: Pole balancing

Goal: Keep the cart on the track and the pole hinged on it away from falling down.



Can be modeled in both ways:

- **Episodic:** +1 reward per time step without failure.
- **Continuous:** -1 discounted reward for failure, 0 otherwise.

Recursiveness of Return

$$
\begin{aligned}
G_t &\triangleq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\
&\triangleq R_{t+1} + \gamma \underbrace{\left[R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots \right]}_{G_{t+1}} \\
&\triangleq R_{t+1} + \gamma G_{t+1}.
\end{aligned}
$$

Makes the divide-and-conquer strategy applicable to RL.

The state-value function

$$v(s) = \mathbb{E}[G_t | S_t = s]$$

Where does $\mathbb{E}[\cdot]$ originate from? What is the source of stochasticity here?

The state-value function

$$v(s) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots]$$

- Expectation needs to be taken over all random variables

$$S_1, S_2, \cdots, R_1, R_2, \cdots$$

- The problem is that we have infinitely many of them!
  - **Markov:** State transitions follow the Markov property.
  - **Reward:** We model rewards as random variables.
  - **Process:** We have a collection of potentially unlimited set of random variables (i.e. a stochastic process).

The state-value function

$$v(s) = \sum_{S_t} \sum_{S_{t+1}} \cdots \sum_{R_{t+1}} \sum_{R_{t+2}} \cdots \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \right]$$

The Bellman Equation

Shows how the recursiveness of return can be manipulated.

$$\begin{aligned} v(s) &= \mathbb{E}[G_t|S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= \underbrace{\mathbb{E}[R_{t+1}|S_t = s]}_{\langle r_s \rangle} + \gamma \mathbb{E}[G_{t+1}|S_t = s]. \end{aligned}$$

Let us take a closer look at the expectation in the second term

$$\mathbb{E}[G_{t+1}|S_t = s] = \sum_{s' \in \mathcal{S}} P[S_{t+1} = s'|S_t = s] \underbrace{\mathbb{E}[G_{t+1}|S_t = s, S_{t+1} = s']}_{v(s')}.$$

Then we get

$$v(s) = \mathbb{E}[R_{t+1}|S_t] + \gamma \sum_{s' \in \mathcal{S}} \underbrace{P[S_{t+1} = s'|S_t = s]}_{P_{ss'}} v(s').$$

The vectorized Bellman Equation

Let us repeat the Bellman equation for all possible states and store the outcomes into a vector

$$\mathbf{v} = \begin{bmatrix} v(s=1) \\ v(s=2) \\ \vdots \\ v(s=n) \end{bmatrix} = \begin{bmatrix} \langle r_1 \rangle + \gamma \sum_{s' \in \mathcal{S}} P_{1s'} v(s') \\ \langle r_2 \rangle + \gamma \sum_{s' \in \mathcal{S}} P_{2s'} v(s') \\ \vdots \\ \langle r_n \rangle + \gamma \sum_{s' \in \mathcal{S}} P_{ns'} v(s') \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} \langle r_1 \rangle \\ \langle r_2 \rangle \\ \vdots \\ \langle r_n \rangle \end{bmatrix}}_{\langle \mathbf{r} \rangle} + \gamma \underbrace{\begin{bmatrix} \sum_{s' \in \mathcal{S}} P_{1s'} v(s') \\ \sum_{s' \in \mathcal{S}} P_{2s'} v(s') \\ \vdots \\ \sum_{s' \in \mathcal{S}} P_{ns'} v(s') \end{bmatrix}}_{\mathbf{Pv}}$$

$$= \langle \mathbf{r} \rangle + \gamma \mathbf{P} \mathbf{v},$$

where $\mathbf{P}$ is the transition matrix with $\mathbf{P}[s, s'] = P_{ss'}$.

Solving the Bellman Equation for the Value Function

$$\begin{aligned} \mathbf{v} &= \langle \mathbf{r} \rangle + \gamma \mathbf{P} \mathbf{v} \\ \mathbf{v}(\mathbf{I} - \gamma \mathbf{P}) &= \langle \mathbf{r} \rangle \\ \mathbf{v} &= (\mathbf{I} - \gamma \mathbf{P})^{-1} \langle \mathbf{r} \rangle \end{aligned}$$

- Has complexity $O(n^3)$ for n states.
- Hence needs to be approximated for many real-world applications.
- How to approximate is a large portion of the remaining course material!

The Markov Decision Process

A tuple of five entities $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where

- $\mathcal{S}$ is the set of environment states: $S_t = s$ with $s \in \mathcal{S}, \ \forall t$.
- $\mathcal{A}$ is the set of actions: $A_t = a$ with $a \in \mathcal{A}, \ \forall a$.
- $\mathcal{R}$ is the set of rewards: $R_t = r$ with $r \in \mathcal{R}, \ \forall r$.
- $\gamma \in [0, 1]$ is the **discount factor**.
- $\mathcal{P} = P(R_{t+1}, S_{t+1} | S_t, A_t)$ is the **environment dynamics model** that naturally decomposes according to the chain rule as

$$P(R_{t+1}, S_{t+1} | S_t, A_t) = \underbrace{P(R_{t+1} | S_t, A_t)}_{Reward\ model} \times \underbrace{P(S_{t+1} | S_t, A_t)}_{transition\ model}.$$

We keep the assumption that the state transitions follow the Markov property

$$P(S_{t+1} | S_t) = P(S_{t+1} | S_1, \cdots, S_t).$$

Policy

Mind that we thus far had a new random variable A_t without an assigned distribution. That distribution is the **policy**, which is defined as a mapping from states to actions

$$\pi(A_t|S_t) = P(A_t|S_t).$$

- MDP models the environment.
- Policy models the agent.
- Our primary concern will be *stationary* policies

$$A_t \sim \pi(\cdot|S_t), \forall t > 0,$$

which behave invariantly of time.

MDP as a MRP

We will typically have an MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ with an attached policy π. Integrate out all actions wrt the policy distribution

$$P^{\pi}_{s,s'} = \sum_{a \in \mathcal{A}} \pi(A_t = a | S_t = s) P(S_{t+1} = s' | S_t = s, A_t = a),$$
$$r^{\pi}_{s} = \sum_{a \in \mathcal{A}} \pi(A_t = a | S_t = s) P(R_{t+1} | S_t = s, A_t = a).$$

Similarly to the MRP case, we construct a transition matrix and a reward vector by evaluating r^{π}_{s} for all states and $P^{\pi}_{s,s'}$ for all state pairs:

$$\mathbf{P}^{\pi}[s, s'] = P^{\pi}_{s,s'}, \qquad \mathbf{r}^{\pi}[s] = r^{\pi}_{s}.$$

We finally attain an MRP with $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}^{\pi}, \mathcal{R}, \gamma \rangle$.

Value Functions in MDPs

State-value function (a.k.a. value function) is the expected return of starting from state s and following policy π

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s].$$

(Intuitively, a measure of how good it is to be in a state)

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

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

(Intuitively, a measure of how good it is to take a certain action in a certain state)

Relationship between state and action value functions

Integrating out the action variable in the action-value function wrt the policy distribution gives the state-value function

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(A_t = a | S_t = s) q_\pi(s, a).$$

Bellman *Expectation* Equation

For the state-value function

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s], \\
&= \sum_{a \in \mathcal{A}} \pi(A_t = a | S_t = s) \Big(\langle r_s^a \rangle + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a v_\pi(s') \Big)
\end{aligned}
$$

where $\langle r_s^a \rangle = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$.

For the action-value function

$$
\begin{aligned}
q_\pi(s,a) &= \mathbb{E}[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a] \\
&= \langle r_s^a \rangle + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a v_\pi(s') \\
&= \langle r_s^a \rangle + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(A_{t+1} = a' | S_{t+1} = s') q_\pi(s', a').
\end{aligned}
$$

Bellman *Expectation* Equation

$$\mathbf{v} = (\mathbf{I} - \gamma \mathbf{P}^{\pi})^{-1} \langle \mathbf{r}^{\pi} \rangle$$

Remark: The solution is a function of $\pi(\cdot|\cdot)$!

Optimal Value Functions and Policies

The optimal state-value function is defined as

$$v_*(s) = \underset{\pi}{\operatorname{argmax}}\ v_\pi(s),$$

and the optimal action-value function as

$$q_*(s,a) = \underset{\pi}{\operatorname{argmax}}\ q_\pi(s,a).$$

In order to maximize subject to policies, we need to define an ordering between them

$$\pi \geq \pi', \quad \text{if} \quad v_\pi(s) \geq v_{\pi'}(s),\ \forall s.$$

Existence of an Optimal Policy

Theorem. For any MDP,

- there exists $\pi_* \geq \pi$, $\forall \pi$.
- optimal policy achieves the optimal value functions: $v_{\pi_*}(s) = v_*(s)$ and $q_{\pi_*}(s, a) = q_*(s, a)$.

Finding an Optimal Policy

Given $q_*(s, a)$, we can find a *deterministic* optimal policy by

$$\pi_*(A_t = a | S_t = s) = \begin{cases} 1, & \text{if } a = \underset{a \in \mathcal{A}}{\text{argmax}}\ q_*(s, a), \\ 0, & \text{otherwise.} \end{cases}$$

That is why the action-value function exists.

Bellman *Optimality* Equation

For the state-value function

$$\begin{aligned} v_*(s) &= \operatorname*{argmax}_a q_*(s,a) \\ &= \operatorname*{argmax}_a \Big[\langle r_s^a \rangle + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a v_*(s') \Big], \end{aligned}$$

and for the action-value function

$$q_*(s,a) = \langle r_s^a \rangle + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a \operatorname*{argmax}_{a'} q_*(s',a').$$

Note the cyclic dependency between $v_*(\cdot)$ and $q_*(\cdot,\cdot)$. We will exploit this fact later.

Finding the optimal policy for an MDP

Highly non-linear problem without closed-form solution.
Iterative alternatives include

- Value iteration
- Policy iteration
- Q-learning
- SARSA

Partially Observable MDPs

Known as POMDPs. The case when the environment cannot be accurately observed.

Defined as a tuple of six entities $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where

- $\mathcal{S}$ is the set of environment states: $S_t = s$ with $s \in \mathcal{S},\ \forall t$.
- $\mathcal{A}$ is the set of actions: $A_t = a$ with $a \in \mathcal{A},\ \forall a$.
- $\mathcal{R}$ is the set of rewards: $R_t = r$ with $r \in \mathcal{R},\ \forall r$.
- $\mathcal{O}$ is the set of observations: $O_t = o$ with $o \in \mathcal{O},\ \forall o$.
- $\gamma \in [0, 1]$ is the **discount factor**.
- $\mathcal{P} = P(R_{t+1}, O_{t+1}, S_{t+1} | S_t, A_t)$ is the **environment dynamics model** that naturally decomposes according to the chain rule as

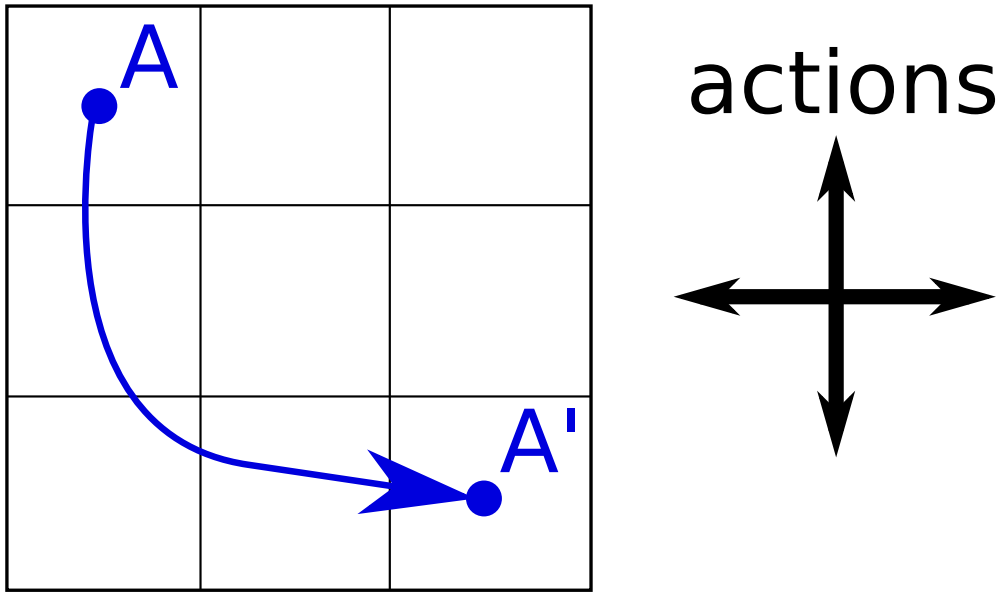
$$P(R_{t+1}, O_{t+1}, S_{t+1} | S_t, A_t) = \underbrace{P(O_{t+1} | S_t)}_{Observation\ model} \underbrace{P(R_{t+1} | S_t, A_t)}_{Reward\ model} \underbrace{P(S_{t+1} | S_t, A_t)}_{transition\ model}.$$

No free lunch theorem for RL

Approximating solutions to MDPs with RL means

- Focus on more frequent cases (state-action pairs)
- at the expense of *very bad* performance on rare cases.

Example: Gridworld



- -1 reward for attempt to go off the grid.
- +10 reward for arriving at A'.
- 0 reward otherwise.

Example: Gridworld

$\langle r_s^a \rangle$	L	U	R	D
00	-1	-1	0	0
01	0	-1	0	0
02	0	-1	-1	0
10	-1	0	0	0
11	0	0	0	0
12	0	0	-1	+10
20	-1	0	0	-1
21	0	0	+10	-1
22	0	0	-1	-1

Choose a good policy

Always go right or down.

$\pi(s\|a)$	L	U	R	D
00	0	0	1/2	1/2
01	0	0	1/2	1/2
02	0	0	0	1
10	0	0	1/2	1/2
11	0	0	1/2	1/2
12	0	0	0	1
20	0	0	1	0
21	0	0	1	1
22	1/2	1/2	0	0

The resultant value function becomes

0	0	0
0	+49.5	+10
0	+100	-

Now choose a bad policy

Go to a random direction.

$\pi(s\|a)$	**L**	**U**	**R**	**D**
00	0	0	1/2	1/2
01	1/3	0	1/3	1/3
02	1/2	0	0	1/2
10	0	1/3	1/3	1/3
11	1/4	1/4	1/4	1/4
12	1/3	1/3	0	1/3
20	0	1/2	1/2	0
21	0	1/3	1/3	1/3
22	1/2	1/2	0	0

The resultant value function becomes

3.35	3.09	2.98
4.34	4.08	3.53
5.19	7.18	4.28