Markov Decision Process

Markov Models

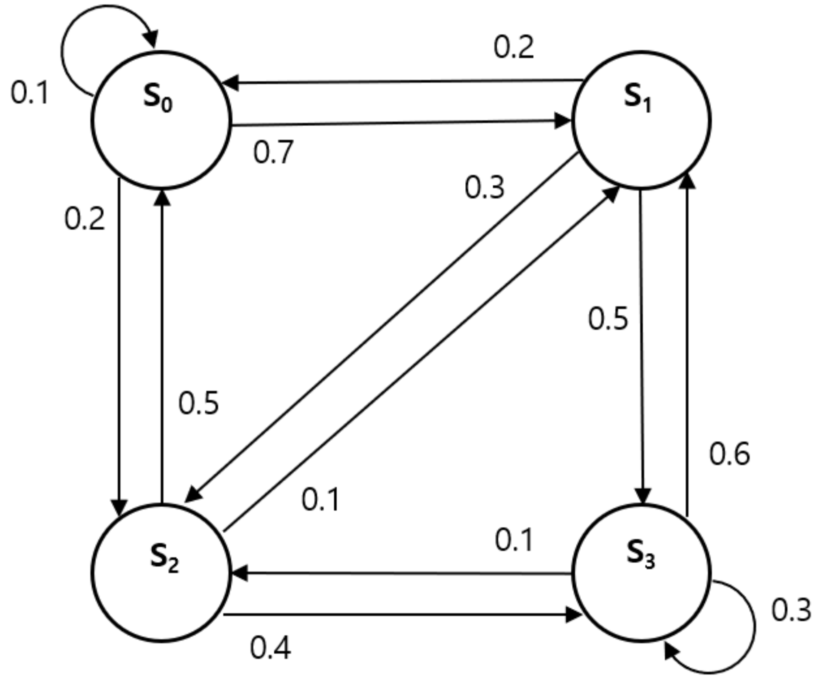
- Markov Model: a stochastic method for randomly changing systems where it is assumed that future states do not depend on past states.

- These models show all possible states as well as the transitions and probabilities between them.

- Different Markov Models
  - Markov Chain(MC)
  - Hidden Markov Model(HMM)
  - Markov Decision Process(MDP)
  - Partially Observable Markov Decision Process(POMDP)

Markov Chain

- Finite number of discrete states
- Probabilistic transitions between state
- Next state determined only by the current state
  - Markov property



$$\begin{array}{cc} & \text{to} \\ \text{from} & \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix} \end{array}$$

	S_0	S_1	S_2	S_3
S_0	0.1	0.7	0.2	
S_1	0.2		0.3	0.5
S_2	0.5	0.1		0.4
S_3		0.6	0.1	0.3

(S_0, S_1, S_2, S_3), (S_0, S_0, S_1, S_3), $(S_0, S_2, S_1, S_2, S_3)$

Markov Process/Chain

- A Markov Chain is a tuple <*S*,*P*>
  - *S* is a (finite) set of states
  - *P* is a state transition probability matrix

$$P_{ss'} = P[S_{t+1} = s' | S_t = s]$$

- Sequence of random variables $S_1, S_2, \ldots$
  - i.e. a sequence of random states $S_1, S_2, \ldots$ with the Markov property.

- Markov process is the continuous-time version of a Markov chain
- A Markov chain/process is a memoryless random process

Recap: Markov Property

- A state S_t is Markov (or information) state if and only if
$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, \ldots S_t]$$

- The future is independent of the past given the present
  - State S_{t+1} is only depending on S_t

- A given system can be fully described by S_t
- Once the state is known, the history may be thrown away
  - i.e. The state is a sufficient statistic of the future
  - Further past observations $S_{t-1}, S_{t-2}, \ldots$ are irrelevant.

State Transition Matrix

- For a Markov state *s* and successor state s', the state transition probability($P_{ss'}$) is defined by

$$P_{ss'} = P[s_{t+1} = s' | S_t = s]$$

- State transition matrix P defines transition probabilities from all states s to all successor states s'

$$P_{ss'} = \begin{array}{cc} & \textit{to} \\ \textit{from} & \begin{bmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nn} \end{bmatrix} \end{array}$$

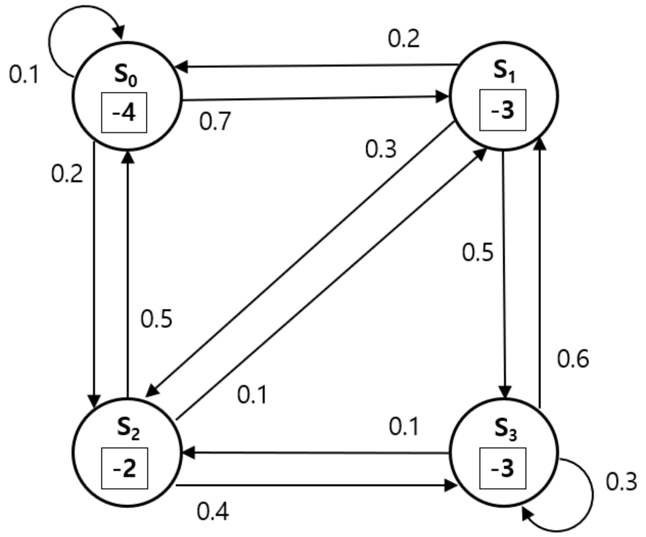
where p_{ij} being the specific probability to go from state i to j

- $\sum_j p_{ij}$=1, $\forall i$

Markov Reward Process

- A Markov Reward Process(MRP) is a tuple $< S, P, R, \gamma >$
  - S is a finite set of states
  - P is a state transition probability matrix, $P_{ss'} = P[S_{t+1} = s' | S_t = s]$
  - R is a *reward function*, $R_t = E[R_{t+1} | S_t = s_t]$
  - γ is a *discount factor*, $\gamma \in [0,1]$

- Markov process extended with rewards
- Still an autonomous stochastic process without specific actions
- Again, rewards R_t only dependent on state S_t



(S_0, S_1, S_2, S_3): $G_0 = -4 + 0.8 * (-3) + 0.8^2 * (-2) + 0.8^3 * (-3) = -9.2$
(S_0, S_0, S_1, S_3): $G_0 = -4 + 0.8 * (-4) + 0.8^2 * (-3) + 0.8^3 * (-3) = -10.6$
(S_0, S_2, S_1, S_2, S_3): $G_0 = -4 + 0.8 * (-2) + 0.8^2 * (-3) + 0.8^3 * (-2) + 0.8^4 * (-3) = -9.7$

Recap: Return

- The return G_t is the total discounted reward from time-step *t*

  $G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{N} \gamma^k R_{t+k+1}$ (finite horizon)
  $G_t = R_{t+1} + \gamma R_{t+2} + \cdots = R_{t+1} + \gamma G_{t+1} = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$ (infinite horizon)

  (*) $G_t = R_{t+1} + \gamma G_{t+1}$

- The discount $\gamma \in [0,1]$ is the present value of future rewards
- The value of receiving reward *R* after k+1 time-steps is $\gamma^k R$
- This values immediate reward above delayed reward.
  - γ close to 0 leads to “short-sighted" evaluation
  - γ close to 1 leads to "far-sighted" evaluation

Why Discount?

- Most Markov reward and decision processes are discounted. Why?
  - Uncertainty about the future may not be fully represented
    - If the reward is financial, immediate rewards may earn more interest than delayed rewards
    - Animal/human behavior shows preference for immediate reward
  - Avoids infinite returns in cyclic Markov processes

- It is sometimes possible to use undiscounted Markov reward processes (i.e. $\gamma = 1$), e.g. if all sequences terminate.

State Value Function

- The state-value function $v(s)$ of an MRP is the expected return starting from state *s*
  - G_t is the total discounted reward from time-step *t*
  - Represents the long-term value of being in state *t*

$$v(s) = E[G_t | S_t = s]$$

  - From 3 paths in p. 7

$$v(s_0) = (-9.2 - 10.6 - 9.7)/3 = -9.83$$

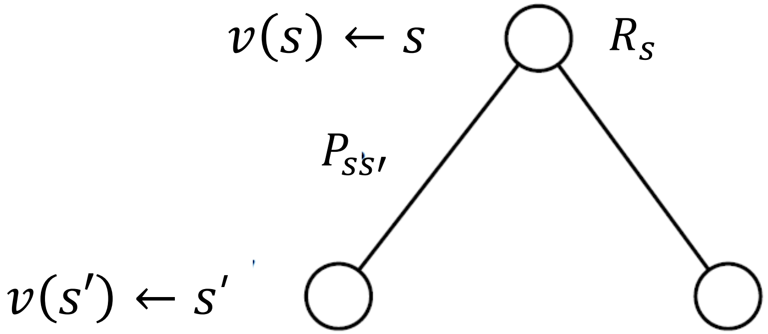
- Problem: How to calculate all state values in closed form?
  - Solution: Bellman equation.

Bellman Equation for MRPs

- The state value function can be decomposed into two parts:
  - immediate reward R_{t+1}
  - discounted value of successor state $\gamma v(s_{t+1})$

$$v(s) = E[G_t | S_t = s] = E[R_{t+1} + \gamma v(s_{t+1}) | S_t = s]$$

  - (changing notation R_{t+1}->R_s & s_{t+1}-> s')



$$\boxed{v(s) = R_s + \gamma \sum_{s' \in \boldsymbol{S}} P_{ss'} v(s')}$$

Bellman Equation in Matrix Form

- For ***v*** being the vector of values $v(s)$, ***R*** being vector in same space of R_s, ∀s ∈ S, and ***P*** being the state transition matrix.
- The Bellman equation can be expressed concisely using matrices,

$$\boldsymbol{v} = \boldsymbol{R} + \gamma \boldsymbol{P} \boldsymbol{v}$$

where $\boldsymbol{v}$ is a column vector with one entry per state

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} R_1 \\ \vdots \\ R_n \end{bmatrix} + \gamma \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & \vdots & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

Solving the Bellman Equation

- It can be solved directly:

$$\boldsymbol{v} = \boldsymbol{R} + \gamma \boldsymbol{P} \boldsymbol{v}$$
$$(\boldsymbol{I} - \gamma \boldsymbol{P})\boldsymbol{v} = \boldsymbol{R}$$
$$\boldsymbol{v} = (\boldsymbol{I} - \gamma \boldsymbol{P})^{-1} \boldsymbol{R}$$

- Computational complexity is $O(n^3)$ for n states
- Direct solution only possible for small MRPs
- There are many iterative methods for large MRPs, e.g.
  - Dynamic programming
  - Monte-Carlo evaluation
  - Temporal-Difference learning

Iterative Algorithm for Computing Value of a MRP

- Dynamic programming

- Initialize $v_0(s) = 0$ for all s
- For k = 1 until convergence (synchronous backups)
  - For all s in S

$$v_k(s) = R_s + \gamma \sum_{s' \in \boldsymbol{S}} P_{ss'} v_{k-1}(s')$$

- Computational complexity: $O(N^2)$ for each iteration

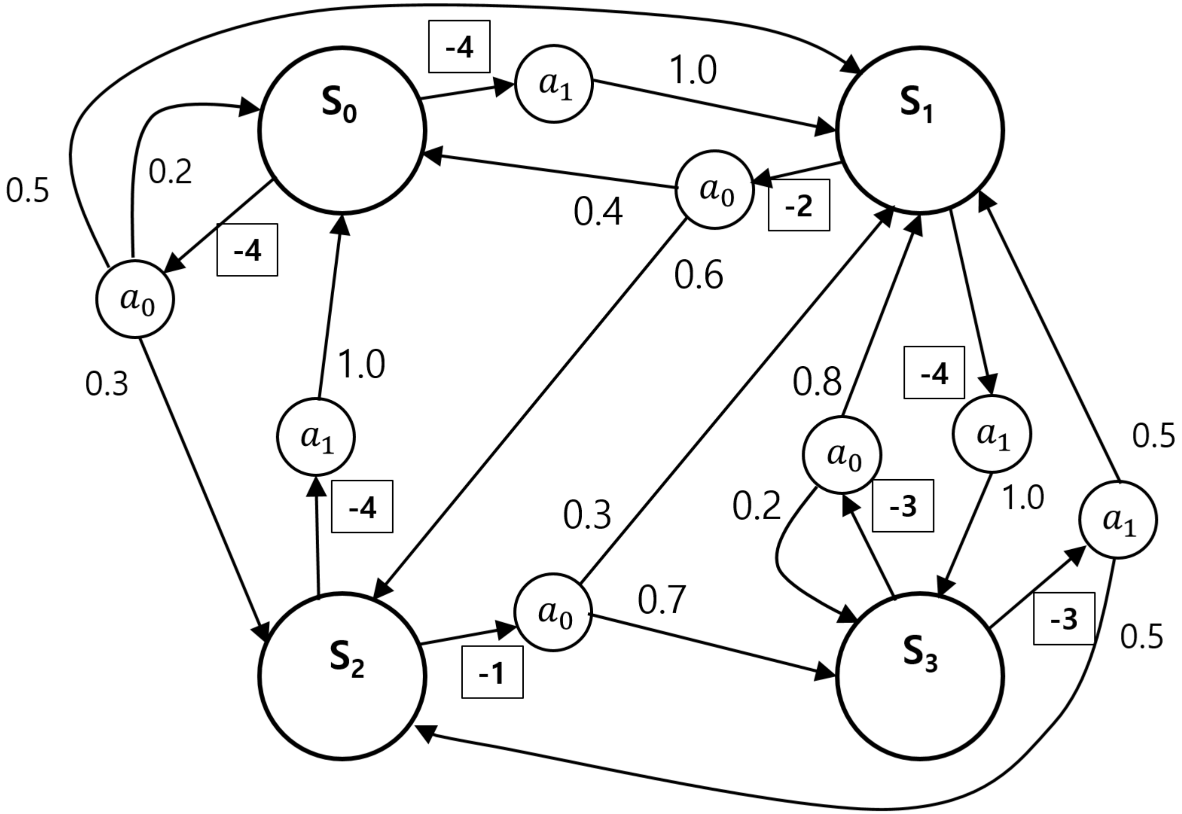
Markov Decision Process

- A Markov decision process (MDP) is a Markov Reward Process with actions.

- A Markov Decision Process is a tuple $< S, A, P, R, \gamma >$
  - S is a finite set of states
  - A is a finite set of actions
  - P is a state transition probability matrix,
    $$P_{ss'}^{a} = P[S_{t+1} = s' | S_t = s, A_t = a]$$
  - R is a reward function,
    $$R_s^a = E[R_{t+1} | S_t = s, A_t = a]$$
  - γ is a discount factor, $\gamma \in [0,1]$

Markov Decision Process

- Basic model of reinforcement learning
- Example of Markov decision process (MDP)



Policies

- A policy fully defines the behavior of an agent
- Policy specifies what action to take in each state
  - Can be deterministic or stochastic

- A policy π is a distribution over actions given states

  - $\pi(a|s) = P[A_t = a|S_t = s]$ (stochastic)
  - $\pi(s) = a$ (deterministic)

- MDP policies depend on the current state (not the history)
  - Policies are stationary (time-independent)

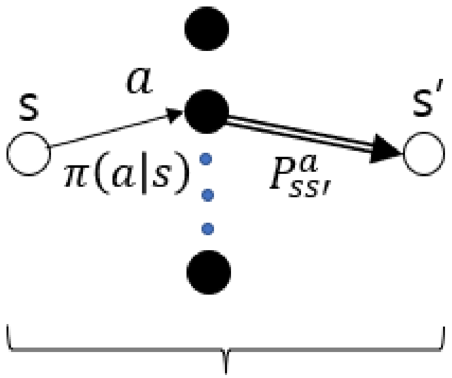
MDP vs MRP

- Given MDP M=$< S, A, P, R, \gamma >$ & a policy π
- The state sequence $S_0, S_1, \ldots$ is a Markov process $< S, P^{\pi}_{ss'} >$
- The state and reward sequence $S_0, R_1, S_1, \ldots$ is a Markov Reward Process $< S, P^{\pi}_{ss'}, R^{\pi}_{s}, \gamma >$ where

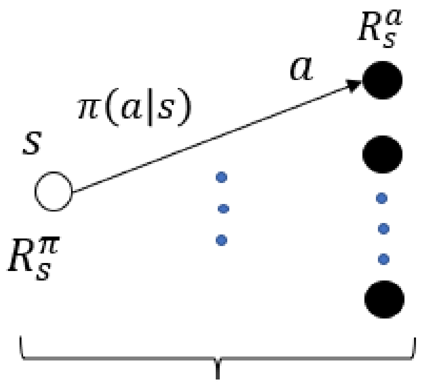
policy π transition prob.

$$P^{\pi}_{ss'} = \sum_{a \in A} \pi(a|s) P^{a}_{ss'}$$

$$R^{\pi}_{s} = \sum_{a \in A} \pi(a|s) R^{a}_{s}$$



$$P^{\pi}_{ss'} = \sum_{a \in A} \pi(a|s) P^{a}_{ss'}$$



$$R^{\pi}_{s} = \sum_{a \in A} \pi(a|s) R^{a}_{s}$$

Value Function

- The *state-value function* $v_\pi(s)$ of an MDP is the expected return starting from state *s* and then following policy π
  - In finite MDPs the state value *v* can be directly linked to the action value *q*
  - (G_t is the total discounted reward from time-step *t*)

$$v_\pi(s_t) = E_\pi[G_t|S_t = s] = \sum_a \pi(a|s) q_\pi(s,a)$$

- The *action-value function* $q_\pi(s,a)$ is the expected return starting from state *s*, taking action *a*, and then following policy π

$$q_\pi(s_t, a_t) = E_\pi[G_t|S_t = s, A_t = a] = R_s^a + \gamma \sum_{s\prime \in S} P_{ss'}^a v_\pi(s')$$

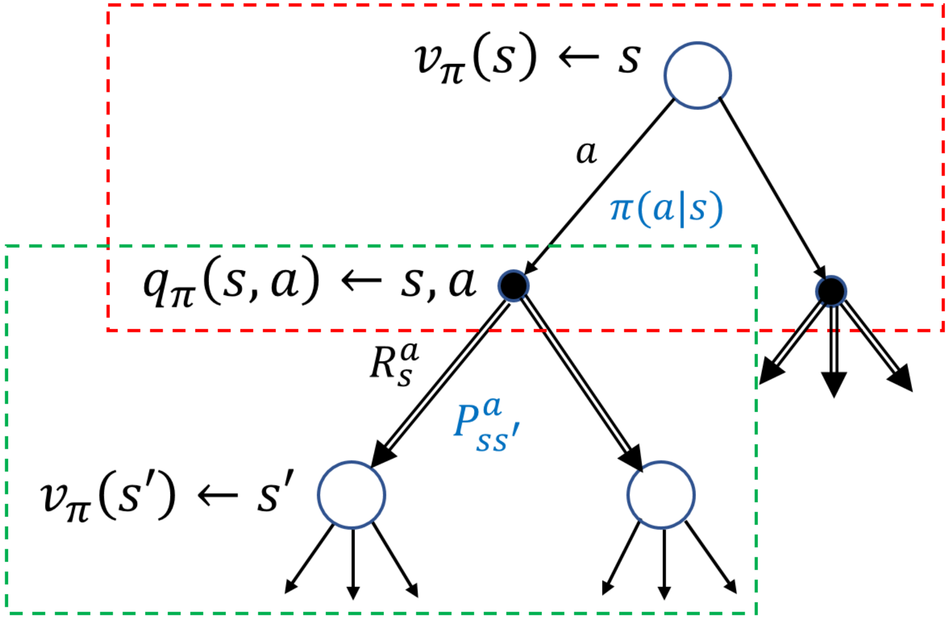
Bellman Expectation Equation for V_π

- In finite MDPs the state value *v* can be directly linked to the action value *q*

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s,a)$$

- Since $q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s')$

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$$
$$= E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1} = s') | S_t = s]$$



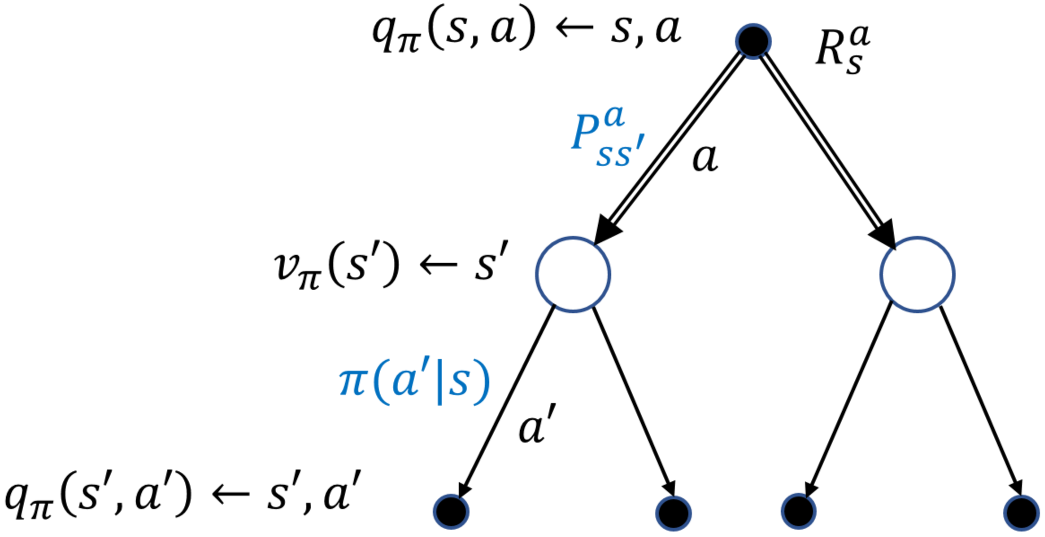
Bellman Expectation Equation for Q_π

- Again, the action value *q* can be directly linked to the state value *v*

$$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s')$$

- Since $v_\pi(s') = \sum_{a' \in A} \pi(a'|s') q_\pi(s',a')$

$$\boxed{q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a'|s') q_\pi(s',a')}$$



Solving the Bellman Expectation Equation

- Bellman equation in MRP (matrix form): $\boldsymbol{v} = \boldsymbol{R} + \gamma \boldsymbol{P} \boldsymbol{v}$

- MDP + Fixed policy -> Markov Reward Process(MRP)
- Given a policy, the MDP reduces to a Markov Reward Process with

$$\boldsymbol{P}^{\boldsymbol{\pi}}_{\boldsymbol{ss'}} = \sum_{a\in A} \pi(a|s) P^{a}_{ss'} \;\&\; \boldsymbol{R}^{\boldsymbol{\pi}}_{\boldsymbol{s}} = \sum_{a\in A} \pi(a|s) R^{a}_{s}$$

- Given a policy π, $P^{\pi}_{ss'}$ and R^{π}_{s} , the Bellman expectation equation (in MDP) can be expressed in Matrix form:

$$v_\pi(s) = \sum_{a\in A} \pi(a|s)\, R^a_s + \gamma \sum_{s'\in S} \sum_{a\in A} \pi(a|s)\, P^a_{ss'} v_\pi(s'))$$

$$\boldsymbol{v_\pi} = \boldsymbol{R^\pi} + \gamma \boldsymbol{P^\pi v_\pi}$$

$$\begin{bmatrix} v_\pi(1) \\ \vdots \\ v_\pi(n) \end{bmatrix} = \begin{bmatrix} R^\pi_1 \\ \vdots \\ R^\pi_n \end{bmatrix} + \gamma \begin{bmatrix} P^\pi_{11} & \cdots & P^\pi_{1n} \\ \vdots & \vdots & \vdots \\ P^\pi_{n1} & \cdots & P^\pi_{nn} \end{bmatrix} \begin{bmatrix} v_\pi(1) \\ \vdots \\ v_\pi(n) \end{bmatrix}$$

Solving the Bellman Expectation Equation

- Here, $\boldsymbol{R}^{\boldsymbol{\pi}}$ and $\boldsymbol{P}^{\boldsymbol{\pi}}$ are the rewards and state transition probability following policy π.

- Hence, the state value can be calculated by solving equation for $\boldsymbol{v}^{\boldsymbol{\pi}}$, e.g. by direct matrix inversion:

$$\boldsymbol{v}_{\boldsymbol{\pi}} = \boldsymbol{R}^{\boldsymbol{\pi}} + \gamma \boldsymbol{P}^{\boldsymbol{\pi}} \boldsymbol{v}_{\boldsymbol{\pi}}$$
$$\boldsymbol{v}_{\boldsymbol{\pi}} = (\boldsymbol{I} - \gamma \boldsymbol{P}^{\boldsymbol{\pi}})^{-1} \boldsymbol{R}^{\boldsymbol{\pi}}$$

Optimal Value Function

- The optimal state-value function $v_*(s)$ is the maximum state value function over all policies

$$v_*(s) = \max_{\pi} v_\pi(s)$$

- The optimal action-value function $q_*(s,a)$ is the maximum action-value function over all policies

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

- The optimal value function denotes the best possible agent's performance for a given MDP / environment.
- A (finite) MDP can be easily solved in an optimal way if $q_*(s,a)$ is known.

Optimal Policy

- What do you mean we say one policy is better than other policy?
- Define a partial ordering over policies

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

- Optimal policies in MDPs:
  - For any MDP, there exists an optimal policy π_* that is better than or equal to all other policies $\pi_* \geq \pi\ , \forall \pi$
  - All optimal policies achieve the optimal state value function, $v_{\pi_*}(s) = v_*(s)$
  - All optimal policies achieve the optimal action-value function, $q_{\pi_*}(s,a) = q_*(s,a)$

Finding an Optimal Policy

- All optimal policy achieve the same optimal state value function and optimal state-action value function

- We find an optimal policy by maximizing over $q_*(s,a)$.
- We solve $q_*(s,a)$ and then we pick the action that gives us most optimal state-action value function

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

- Therefore, *if we know* $q_*(s,a)$ *, we have the optimal policy*
- There is always a deterministic optimal policy for any MDP

Finding an Optimal Policy

- We can also find an optimal policy using $v_*(s, a)$
- However, in this case, we need to know R_s^a & $P_{ss'}^a$ (model-based)

$$\pi_*(a|s) = \begin{cases} 1, & \text{if } a = \text{argmax}_{a \in A} \boldsymbol{R_s^a} + \gamma \boldsymbol{P_{ss'}^a} v_*(s') \\ 0, & \text{otherwise} \end{cases}$$

- In model-free case (R_s^a & $P_{ss'}^a$ not known), we can not find an optimal policy using v_* value

Bellman Optimality Equation

- If we know $q_*(s, a)$, we have the optimal policy.
- Then how do we find these $q_*(s, a)$ values ?

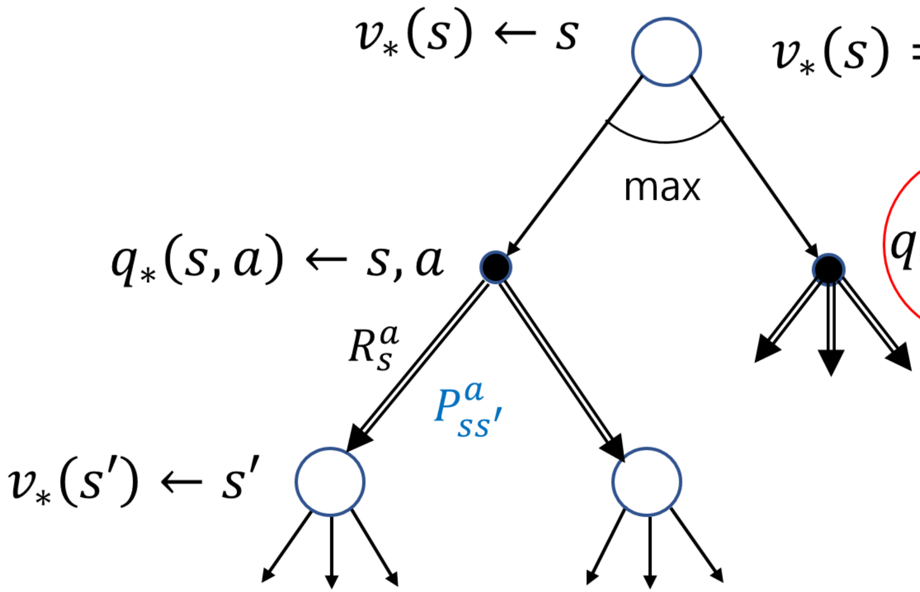
- This is where Bellman Optimality Equation comes into play.

- Bellman Optimality Equation is the same as Bellman Expectation Equation but the only difference is instead of taking the *average* of the actions our agent can take we take the action with the *max* value.

- Bellman Optimality Equation: The Optimal Value Function is recursively related to the Bellman Optimality Equation.

Bellman Optimality Equation for V_*

- Again, the Bellman optimality equation can be visualized by a backup diagram:



$$v_*(s) = \max_a q_*(s,a)$$

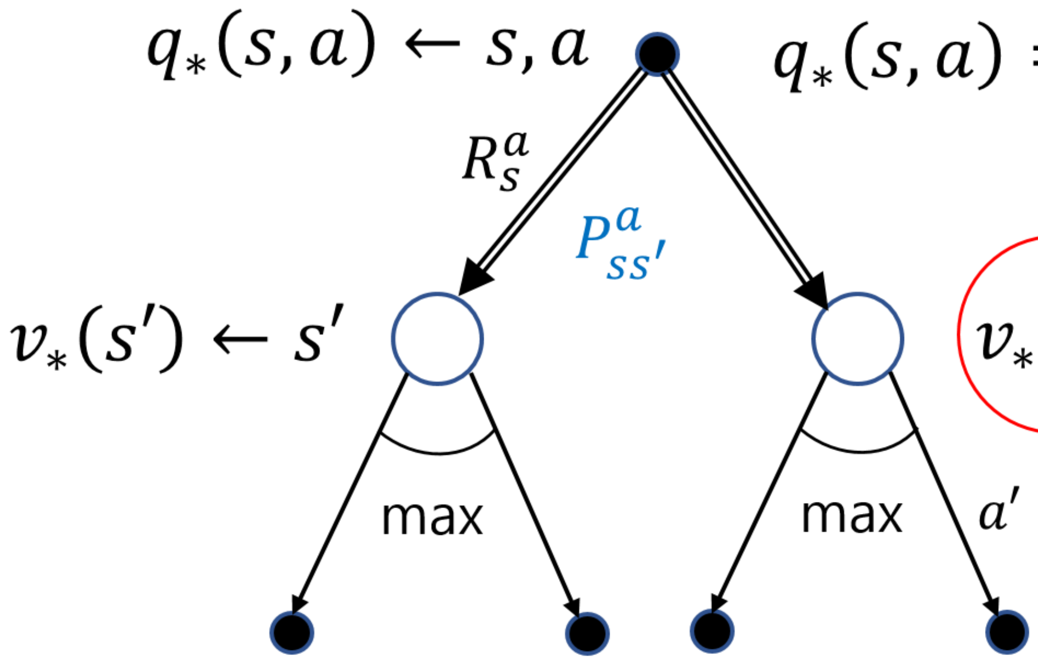
$$q_*(s,a) = R_s^a + \gamma \textstyle\sum_{s' \in S} P_{ss'}^a v_*(s')$$

$$\boxed{v_*(s) = \max_a \left(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_*(s')\right)}$$

- (*) max is not applied in $q_*(s,a)$ since q doesn't contain actions

Bellman Optimality Equation for Q_*

- Likewise, the Bellman optimality equation is applicable to the action value:



$$q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_*(s')$$

$$v_*(s') = \max_{a'} q_*(s',a')$$

$$\boxed{q_*(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \max_{a'} q_*(s',a')}$$

Solving Bellman Optimality Equation

- Can we compute optimal value functions from Bellman Optimality Equation ?

- MRP -> Bellman Equation (matrix) -> direct solution
- MDP + fixed policy -> Bellman Expectation Equation (matrix) -> direct solution
- MDP -> Bellman Optimality Equation -> ?

- Due to *max* operator, the equation set is generally nonlinear. Direct, closed form solution not available (in general).
- Hence, often approximate/iterative solutions are required
- Many iterative solution methods
  - Value Iteration
  - Policy Iteration
  - Q-learning
  - Sarsa