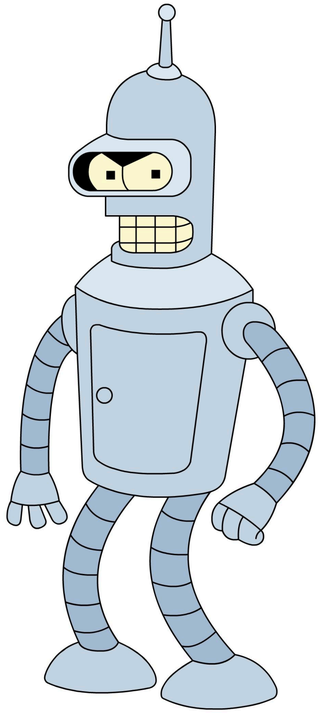
Reinforcement Learning

HSE, autumn - winter 2022

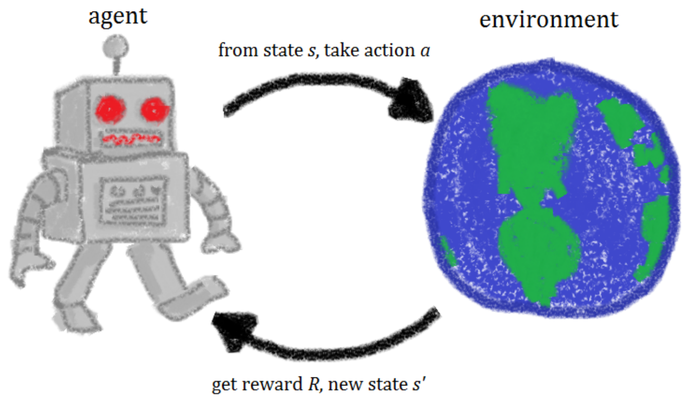
Lecture 2: Dynamic Programming



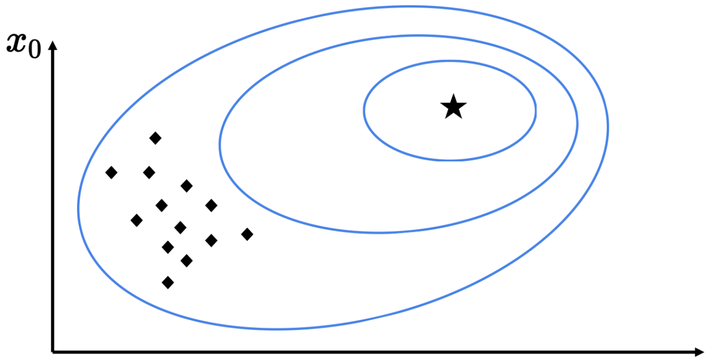
Sergei Laktionov
slaktionov@hse.ru
LinkedIn

Previously on RL Course

- RL formalism: state (observation), action, reward, MDP, cumulative reward etc.
- (Deep) cross-entropy method:
  - Intuitive idea, rich theoretical background, satisfactory results
  - Zeroth-order **black-box** optimization technique
    - No information regarding the environment rewards and dynamics



Source



Source

However if we really know something about the environment…

Background

1. Sutton & Barto, Chapter 3 + 4
2. RL for Finance Book, Chapter 4 + 5
3. Practical RL course by YSDA, week 2
4. Past iteration course, lecture 2
5. DeepMind course, lectures 3 + 4

Markov Decision Process

MDP is a 4-tuple $(\mathcal{S}, \mathcal{A}, p, r)$:

1. $\mathcal{A}$ is an action space
2. $\mathcal{S}$ is a state space
3. $p(s' \mid s, a) = \mathbb{P}(S_{t+1} = s' \mid S_t = s, A_t = a)$ is a state-transition function
4. $r : \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a reward function giving an expected reward:
   $r(s, a) = \mathbb{E}[R \mid s, a]$

Markov property:
(the future is independent of the past given the present)

$$p(R_t, S_{t+1} \mid S_t, A_t, R_{t-1}, S_{t-1}, A_{t-1}, \ldots) = p(R_t, S_{t+1} \mid S_t, A_t)$$

Abuse of Notation

1. R_t, S_t, A_t are random variables;

2. r, s, a are realisation of random variables;

3. $r(s, a)$ is a function which maps state and action realisations into conditional reward expectation.

Equivalent Definition of MDP

MDP is a 3-tuple $(\mathcal{S}, \mathcal{A}, p)$:

1. $\mathcal{A}$ is an action space
2. $\mathcal{S}$ is a state space
3. $p(r, s' \mid s, a) = \mathbb{P}(R_t = r, S_{t+1} = s' \mid S_t = s, A_t = a)$ is the joint probability of a reward r and next state s', given a state s and action a

Markov property:
(the future is independent of the past given the present)

$$p(R_t, S_{t+1} \mid S_t, A_t, R_{t-1}, S_{t-1}, A_{t-1}, \ldots) = p(R_t, S_{t+1} \mid S_t, A_t)$$

Equivalent Definition of MDP

MDP is a 3-tuple $(\mathcal{S}, \mathcal{A}, p)$:

1. $\mathcal{A}$ is an action space
2. $\mathcal{S}$ is a state space
3. $p(r, s' \mid s, a) = \mathbb{P}(R_t = r, S_{t+1} = s' \mid S_t = s, A_t = a)$ is the joint probability of a reward r and next state s', given a state s and action a

$$p(s' \mid s, a) = \sum_r p(s', r \mid s, a) \qquad r(s, a) = \sum_r r \sum_{s'} p(s', r \mid s, a)$$

Relations

1. $p(s' \mid s, a) = \sum_r p(s', r \mid s, a)$

2. $r(s, a) = \sum_r r \sum_{s'} p(s', r \mid s, a)$

3. $r(s, a, s') = \mathbb{E}[R_t \mid S_t = s, A_t = a, S_{t+1} = s'] = \sum_r r\, p(r \mid s, a, s') =$

$$= \sum_r r \frac{p(r, s', s, a)}{p(s', s, a)} = \sum_r r \frac{p(r, s' \mid s, a)}{p(s' \mid s, a)}$$

Returns and Episodes

Let T is a final time step. If $T < \infty$ then environment is called *episodic*.

Cumulative reward is called a return or reward-to-go. Note that in general it is a random variable.

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots = \sum_{s=t}^{T} \gamma^{s-t} R_s$$

Episode length

Immediate reward

Discount factor

Reminder: either $T = \infty$ or $\gamma = 1$

Returns and Episodes

Let T is a final time step. If $T < \infty$ then environment is called *episodic*.

Cumulative reward is called a return or reward-to-go. Note that in general it is a random variable.

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots = \sum_{s=t}^{T} \gamma^{s-t} R_s$$

Episode length

Immediate reward

Discount factor

Important note: $G_t = R_t + \gamma(R_{t+1} + \gamma R_{t+2} + \ldots) = R_t + \gamma G_{t+1}$

Objective Function

Agent interacts with the environment following some policy $\pi(a \mid s)$. As the result a trajectory τ is generated: $\tau = (s_0, a_0, s_1, \ldots, a_{T-1}, s_T), a_t \sim \pi(\,. \mid s_t)$

$$J(\pi) = \mathbb{E}_{\pi}[G_0] = \mathbb{E}_{p(\tau|\pi)}[G_0] =$$

$$\mathbb{E}_{s_0 \sim p(s_0)}[\mathbb{E}_{a_0 \sim \pi(.|s_0)}[R_0 + \mathbb{E}_{s_1 \sim p(.|s_0,a_0)}[\mathbb{E}_{a_1 \sim \pi(.|s_1)}[\gamma R_1 + \ldots]]]] \rightarrow \max_{\pi}$$

State-value Function

Estimate how good it is for an agent to be in a given state s and follow a policy π:

$$V_\pi(s) = \mathbb{E}_\pi[G_t \,|\, S_t = s] = \mathbb{E}_\pi[\sum_{s=0}^{T} \gamma^s R_{t+s} \,|\, S_t = s]$$

If s is a terminal state then $V_\pi(s) = 0$.

State-value Function

Law of iterated expectation

$$V_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi[R_t + \gamma G_{t+1} \mid S_t = s] =$$

$$= \sum_a \pi(a \mid s) \sum_{r,s'} \mathbb{P}(R_t = r, S_{t+1} = s' \mid S_t = s, A_t = a, S_{t-1} = s'', A_{t-1} = a'', R_{t-1} = r'', \ldots)\left[r + \gamma \mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s'\right] =$$

Definition of reward-state transition function

$$= \sum_a \pi(a \mid s) \sum_{r,s'} \mathbb{P}(R_t = r, S_{t+1} = s' \mid S_t = s, A_t = a)[\ldots] =$$

Markov property

Definition of $V^\pi(s')$

$$= \sum_a \pi(a \mid s) \sum_{r,s'} p(r, s' \mid s, a)\left[r + \gamma \mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s']\right]$$

$$= \sum_a \pi(a \mid s) \sum_{r,s'} p(r, s' \mid s, a)\left[r + \gamma V_\pi(s')\right]$$

Policy stochasticity

Environment stochasticity

Action-value Function

Estimate how good it is for the agent to take an action a being in a given state s and follow a policy π:

$$Q_\pi(s,a) = \mathbb{E}_\pi[G_t \,|\, S_t = s, A_t = a] = \mathbb{E}_\pi[\sum_{s=0}^{T} \gamma^s R_{t+s} \,|\, S_t = s, A_t = a]$$

If s is a terminal state then $\forall a \in \mathcal{A}\ Q_\pi(s,a) = 0$.

Action-value Function

$$Q_\pi(s,a) = \mathbb{E}_\pi[G_t \,|\, S_t = s, A_t = a] = \mathbb{E}_\pi[R_t + \gamma G_{t+1} \,|\, S_t = s, A_t = a] =$$

$$= \sum_{r,s'} p(r, s' \,|\, s, a) \Big[r + \gamma \mathbb{E}_\pi[G_{t+1} \,|\, S_{t+1} = s'] \Big]$$

$$= \sum_{r,s'} p(r, s' \,|\, s, a) \Big[r + \gamma V_\pi(s') \Big]$$

Relationship between V_π and Q_π

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{r,s'} p(r, s' \mid s, a) \left[r + \gamma V_\pi(s') \right]$$

$$Q_\pi(s, a) = \sum_{r,s'} p(r, s' \mid s, a) \left[r + \gamma V_\pi(s') \right]$$

Relationship between V_π and Q_π

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{r,s'} p(r, s' \mid s, a) \left[r + \gamma V_\pi(s') \right]$$

$$Q_\pi(s, a) = \sum_{r,s'} p(r, s' \mid s, a) \left[r + \gamma V_\pi(s') \right]$$

$$V_\pi(s) = \sum_a \pi(a \mid s) Q_\pi(s, a) = \mathbb{E}_{a \sim \pi(. \mid s)} Q_\pi(s, a)$$

Relationship between V_π and Q_π

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{r,s'} p(r, s' \mid s, a) \left[r + \gamma V_\pi(s') \right]$$

$$Q_\pi(s, a) = \sum_{r,s'} p(r, s' \mid s, a) \left[r + \gamma V_\pi(s') \right]$$

$$V_\pi(s) = \sum_a \pi(a \mid s) Q_\pi(s, a) = \mathbb{E}_{a \sim \pi(. \mid s)} Q_\pi(s, a)$$

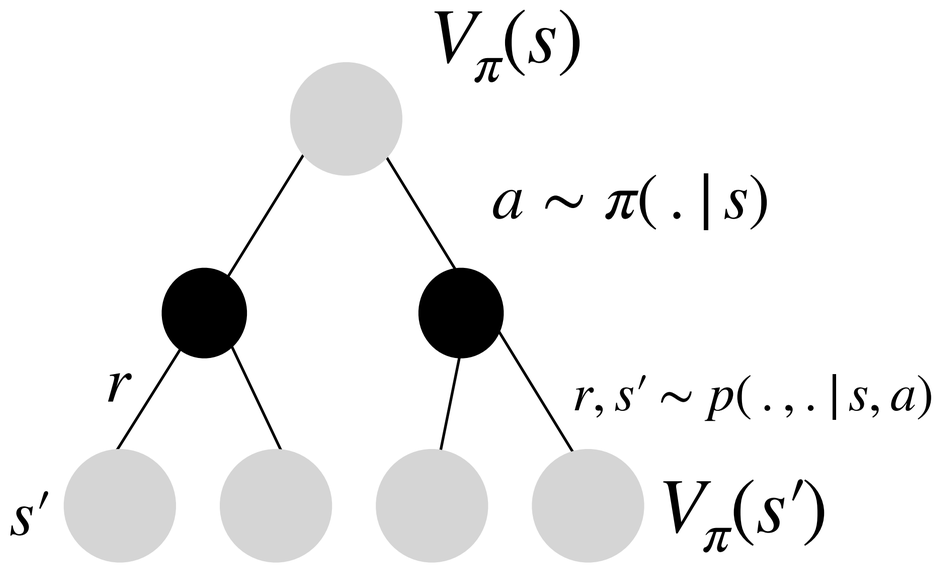
$$Q_\pi(s, a) = \sum_{r,s'} p(r, s' \mid s, a) \left[r + \gamma \sum_{a'} \pi(a' \mid s') Q_\pi(s', a') \right]$$

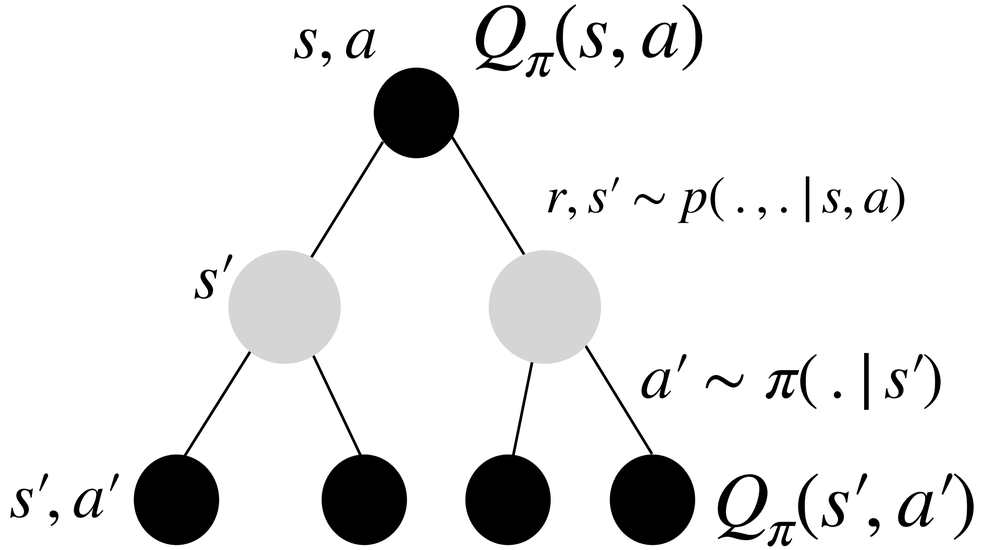
Bellman Expectations Equations

- $V_\pi(s) = \sum_a \pi(a \mid s) \sum_{r,s'} p(r, s' \mid s, a) \left[r + \gamma V_\pi(s') \right] = \mathbb{E}_\pi[R_t + \gamma V_\pi(S_{t+1}) \mid S_t = s]$
- $Q_\pi(s, a) = \sum_{r,s'} p(r, s' \mid s, a) \left[r + \gamma \sum_{a'} \pi(a' \mid s') Q_\pi(s', a') \right] = r(s, a) + \gamma \mathbb{E}_\pi Q(s', a')$

Bellman Expectations Equations

- $V_\pi(s) = \sum_a \pi(a \mid s) \sum_{r,s'} p(r, s' \mid s, a) \left[r + \gamma V_\pi(s') \right] = \mathbb{E}_\pi[R_t + \gamma V_\pi(S_{t+1}) \mid S_t = s]$
- $Q_\pi(s, a) = \sum_{r,s'} p(r, s' \mid s, a) \left[r + \gamma \sum_{a'} \pi(a' \mid s') Q_\pi(s', a') \right] = r(s, a) + \gamma \mathbb{E}_\pi Q(s', a')$





Bellman Optimality Equations

Recall that our goal is to find a policy that achieves maximum expected cumulative reward.

Define a partial order on a set of possible policies:

$$\pi \geq \pi' \iff V_\pi(s) \geq V_{\pi'}(s) \; \forall s \in \mathcal{S}$$

Let π^* be all optimal policies and define optimal value functions:

$$V^*(s) = \max_\pi V_\pi(s)$$

$$\pi^*(a \mid s) = \mathbb{I}[a = argmax_{a'} Q^*(s, a')]$$

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

Bellman Optimality Equations

$$V^*(s) = \max_{\pi} V_{\pi}(s)$$

$$Q^*(s,a) = \max_{\pi} Q_{\pi}(s,a)$$

$$V^*(s) = \max_{a} Q^*(s,a)$$

$$V^*(s) = \max_{a} \mathbb{E}_{\pi^*}[G_t \,|\, s_t = s, a_t = a] = \max_{a} \mathbb{E}_{\pi^*}[R_t + \gamma G_{t+1} \,|\, s_t = s, a_t = a] =$$

$$= \max_{a} \mathbb{E}[R_t + \gamma V^*(s_{t+1}) \,|\, s_t = s, a_t = a] = \max_{a} \sum_{s',r} p(s', r \,|\, s, a)[r + \gamma V^*(s')]$$

Bellman Optimality Equations

$$Q_\pi(s,a) = \sum_{r,s'} p(r,s' \mid s,a)\Big[r + \gamma V_\pi(s')\Big]$$

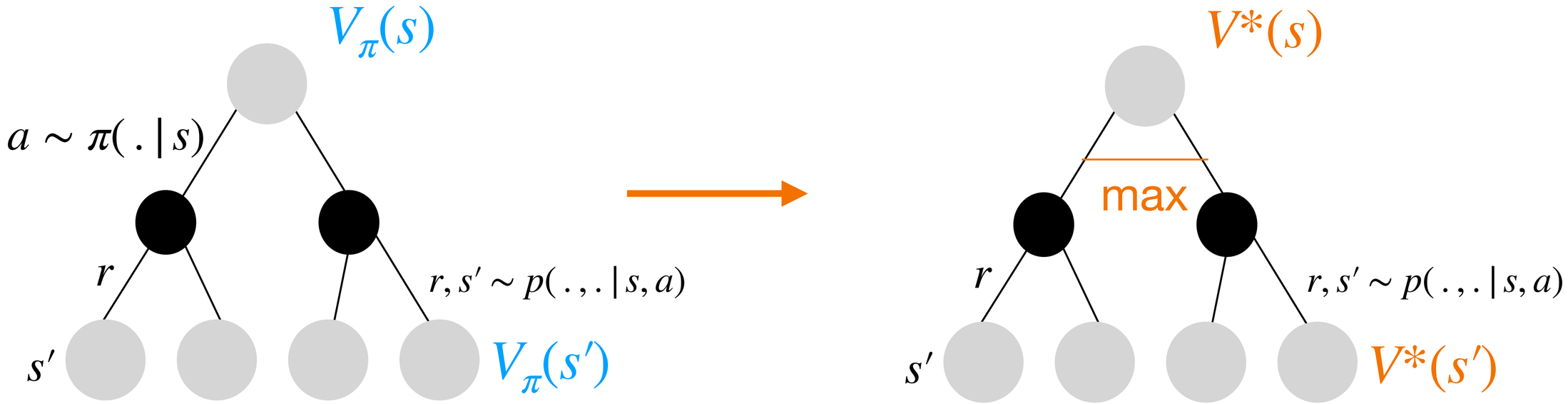
$$Q^*(s,a) = \mathbb{E}[R_t + \gamma V^*(s_{t+1}) \mid s_t = s, a_t = a] = \sum_{s',r} p(s',r \mid s,a)\Big[r + \gamma \max_{a'} Q^*(s',a')\Big]$$

Bellman Optimality Equations

$$V^*(s) = \max_a \mathbb{E}[R_t + \gamma V^*(s_{t+1}) \,|\, s_t = s, a_t = a] = \max_a \sum_{s',r} p(s', r \,|\, s, a)[r + \gamma V^*(s')]$$

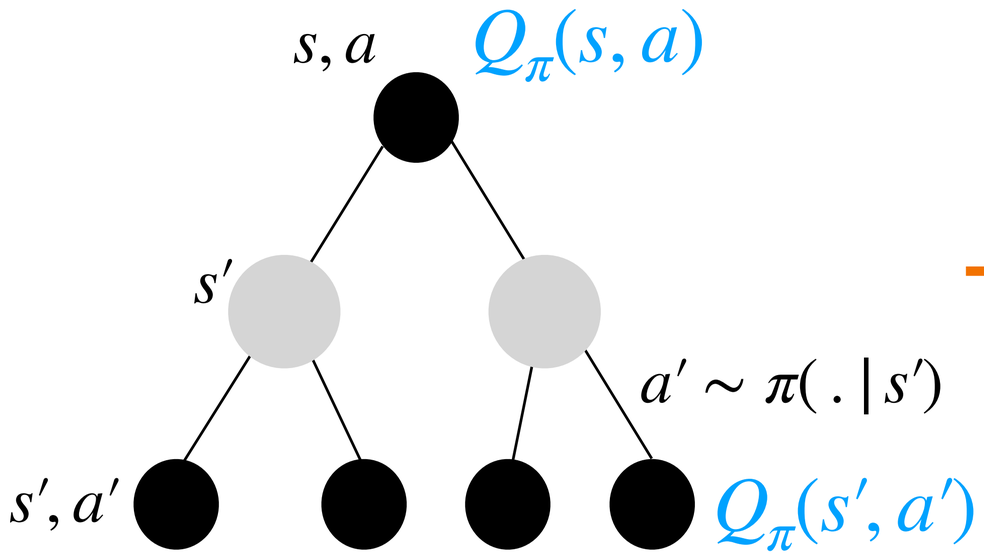
$$Q^*(s, a) = \mathbb{E}[R_t + \gamma V^*(s_{t+1}) \,|\, s_t = s, a_t = a] = \sum_{s',r} p(s', r \,|\, s, a)\left[r + \gamma \max_{a'} Q^*(s', a')\right]$$

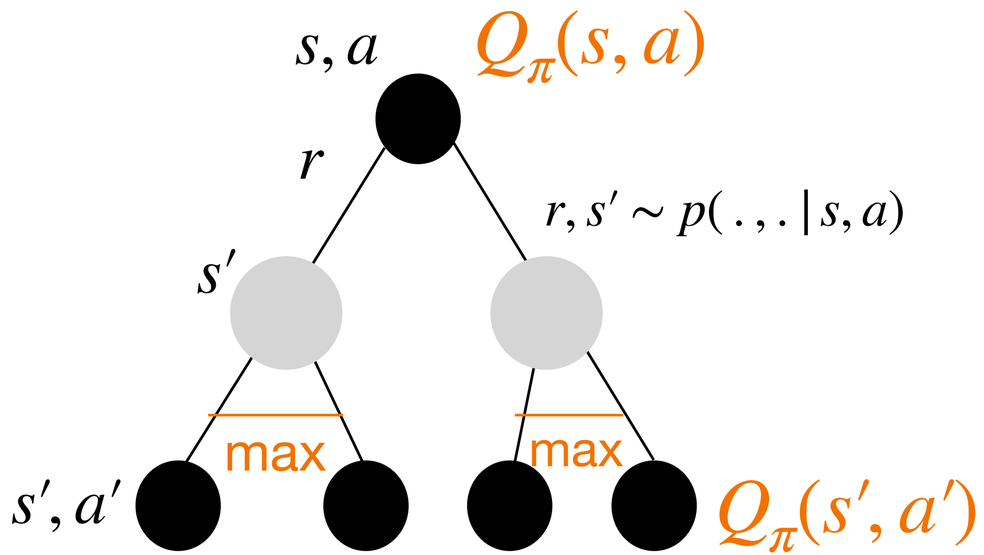
Intuition



$$V^*(s) = \max_a \mathbb{E}[R_t + \gamma V^*(s_{t+1}) \,|\, s_t = s, a_t = a] = \max_a \sum_{s',r} p(s', r \,|\, s, a)[r + \gamma V^*(s')]$$

Intuition



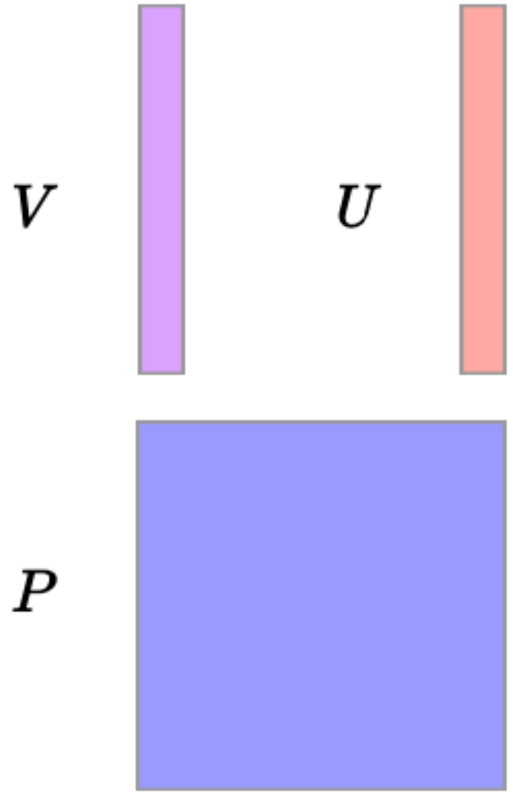


$$Q^*(s,a) = \mathbb{E}[R_t + \gamma V^*(s_{t+1}) \,|\, s_t = s, a_t = a] = \sum_{s',r} p(s', r \,|\, s, a)\Big[r + \gamma \max_{a'} Q^*(s', a')\Big]$$

Linear Equations

$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{r,s'} p(r, s' \mid s, a)\left[r + \gamma V_\pi(s')\right]$ is a system of $|S|$ linear equations:

$$V = U + \gamma PV \rightarrow (I - \gamma P)V = U \rightarrow V = (I - \gamma P)^{-1}U$$



Source

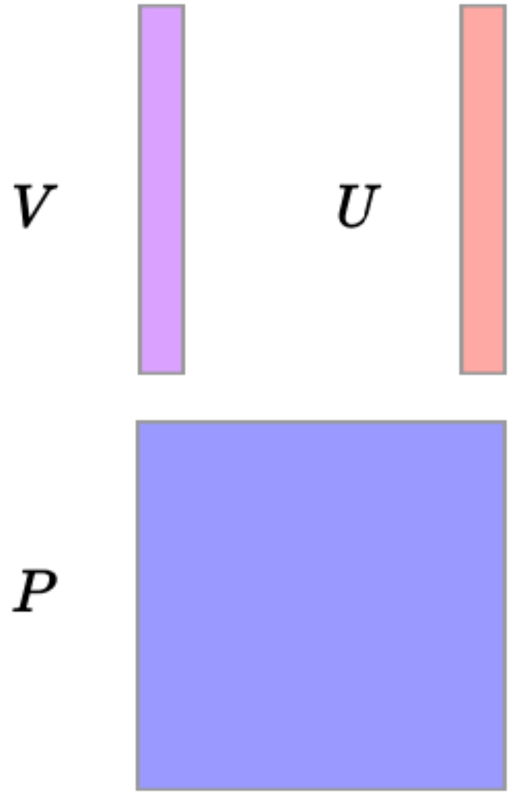
Linear Equations

$V_\pi(s) = \sum_a \pi(a|s) \sum_{r,s'} p(r, s'|s, a)\left[r + \gamma V_\pi(s')\right]$ is a system of $|S|$ linear equations:

$$V = U + \gamma PV \rightarrow (I - \gamma P)V = U \rightarrow V = (I - \gamma P)^{-1}U$$

General case: $\mathcal{O}(|S|^3)$

Maybe we can try iterative solution methods?



Source

Dynamic Programming

Problems in RL

1. Estimating V^{π} or Q^{π} is called policy evaluation or prediction

2. Estimating V^{*} or Q^{*} is called control because they can be used for policy optimisation

Bellman Operators

Bellman expectation operator for $V(s)$:

$$[\mathcal{T}^{\pi}V](s) = \mathbb{E}_{r,s'|s,a\sim\pi(.|s)}[r + \gamma V(s')]$$

Bellman expectation operator for $Q(s,a)$:

$$[\mathcal{T}^{\pi}Q](s,a) = \mathbb{E}_{r,s'|s,a}\Big[r + \gamma \mathbb{E}_{a'\sim\pi(.|s)}[Q(s',a')]\Big]$$

Bellman optimality operator for $V(s)$:

$$[\mathcal{T}V](s) = \max_{a} \mathbb{E}_{r,s'|s,a}[r + \gamma V(s')]$$

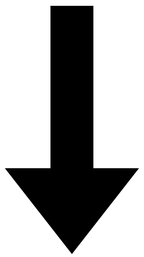
Bellman optimality operator for $V(s)$:

$$[\mathcal{T}Q](s,a) = \mathbb{E}_{r,s'|s,a}\Big[r + \gamma \max_{a'} Q(s',a')\Big]$$

Policy evaluation

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{r,s'} p(r, s' \mid s, a) \left[r + \gamma V_\pi(s') \right]$$

Linear equations



$$V_{k+1}(s) = [\mathcal{T} V_k](s) = \sum_a \pi(a \mid s) \sum_{r,s'} p(r, s' \mid s, a) \left[r + \gamma V_k(s') \right]$$

for fixed policy π

Simple iteration method

Convergence Guarantees

Assume that $\mathcal{S}$, $\mathcal{A}$ are finite.

Convergence Guarantees

Assume that $\mathcal{S}$, $\mathcal{A}$ are finite.

Monotonicity:

Lemma:

For all V, U if $V(s) \leq U(s)\ \forall s \in \mathcal{S}$ then $(\mathcal{T}^{\pi}V)(s) \leq (\mathcal{T}^{\pi}U)(s)\ \forall s \in \mathcal{S}$

Proof:

$$(\mathcal{T}^{\pi}V)(s) = \mathbb{E}_{r,s'|s,a=\pi(s)}\left[r + \gamma V(s')\right] \leq \mathbb{E}_{r,s'|s,a=\pi(s)}\left[r + \gamma U(s')\right] = (\mathcal{T}^{\pi}U)(s)$$

Convergence Guarantees

Contraction:

Theorem: $||\mathcal{T}^{\pi}V - \mathcal{T}^{\pi}U||_{\infty} \leq \gamma ||V - U||_{\infty}$

Proof:

For all V, U:

$$||\mathcal{T}^{\pi}V - \mathcal{T}^{\pi}U||_{\infty} = ||\mathbb{E}_{r,s'|s,a=\pi(s)}\left[r + \gamma V(s')\right] - \mathbb{E}_{r,s'|s,a=\pi(s)}\left[r + \gamma U(s')\right]||_{\infty} =$$

$$= ||\mathbb{E}_{r,s'|s,a=\pi(s)}\left[\gamma V(s') - \gamma U(s')\right]||_{\infty} \leq \gamma \max_{s'} |V(s') - U(s')| = \gamma ||V - U||_{\infty}$$

Convergence Guarantees

Convergence:

By Banach fixed point theorem, $V_k \rightarrow_{||.||_\infty} V_\pi$ such that:

$$[\mathcal{T}^\pi V_\pi](s) = \mathbb{E}_{r,s'|s,a\sim\pi(.|s)}[r + \gamma V_\pi(s')]$$

Policy Evaluation Algorithm

Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

Bellman expectation operator for $V^\pi(s)$

Policy Improvement

Let us act greedily w.r.t. $Q^{\pi}(s,a)$:

$$\pi'(s) = argmax_a Q^{\pi}(s,a) = argmax_a \sum_{r,s'} p(r,s' \mid s,a)[r + \gamma V^{\pi}(s')]$$

Policy Improvement

Let us act greedily w.r.t. $Q^{\pi}(s,a)$:

$$\pi'(s) = argmax_a Q^{\pi}(s,a) = argmax_a \sum_{r,s'} p(r,s'|s,a)[r + \gamma V^{\pi}(s')]$$

π^* is not worse than π:

If $\forall s \in \mathcal{S}\ Q^{\pi}(s,\pi'(s)) \geq V^{\pi}(s)$

then $\forall s \in \mathcal{S}\ V^{\pi'}(s) \geq V^{\pi}(s) \iff \pi' \geq \pi$

Policy Improvement

Let us act greedily w.r.t. $Q^{\pi}(s,a)$:

$$\pi'(s) = argmax_a Q^{\pi}(s,a) = argmax_a \sum_{r,s'} p(r,s' \mid s,a)[r + \gamma V^{\pi}(s')]$$

If $\pi' = \pi \rightarrow V^{\pi'} = V^{\pi}$ and V^{π} satisfies Bellman optimality equation:

$$V^{\pi}(s) = \max_a \sum_{r,s'} p(r,s' \mid s,a)[r + \gamma V^{\pi}(s')]$$

Convergence Guarantees

Convergence:

The convergence to optimal value function V^* is guaranteed by Banach fixed point theorem applied to Bellman optimality operator for $V(s)$:

$$[\mathcal{T}V](s) = \max_a \mathbb{E}_{r,s'|s,a}[r + \gamma V(s')]$$

Due to monotonicity we have a monotonic sequence of policies $\pi_0 \leq \pi_1 \leq \ldots \leq \pi^*$. Because a finite MDP has only a finite number of policies, this process must converge to an optimal policy in a finite number of iterations.

Recover Optimal Policy

1. If Q^* is known: $\pi(s) = argmax_a Q^*(s,a)$
2. If V^* is known: $\pi(s) = argmax_a \sum_{r,s'} p(r,s' \mid s,a)[r + \gamma V^*(s')]$

Unknown in a model-free setting

Policy Iteration

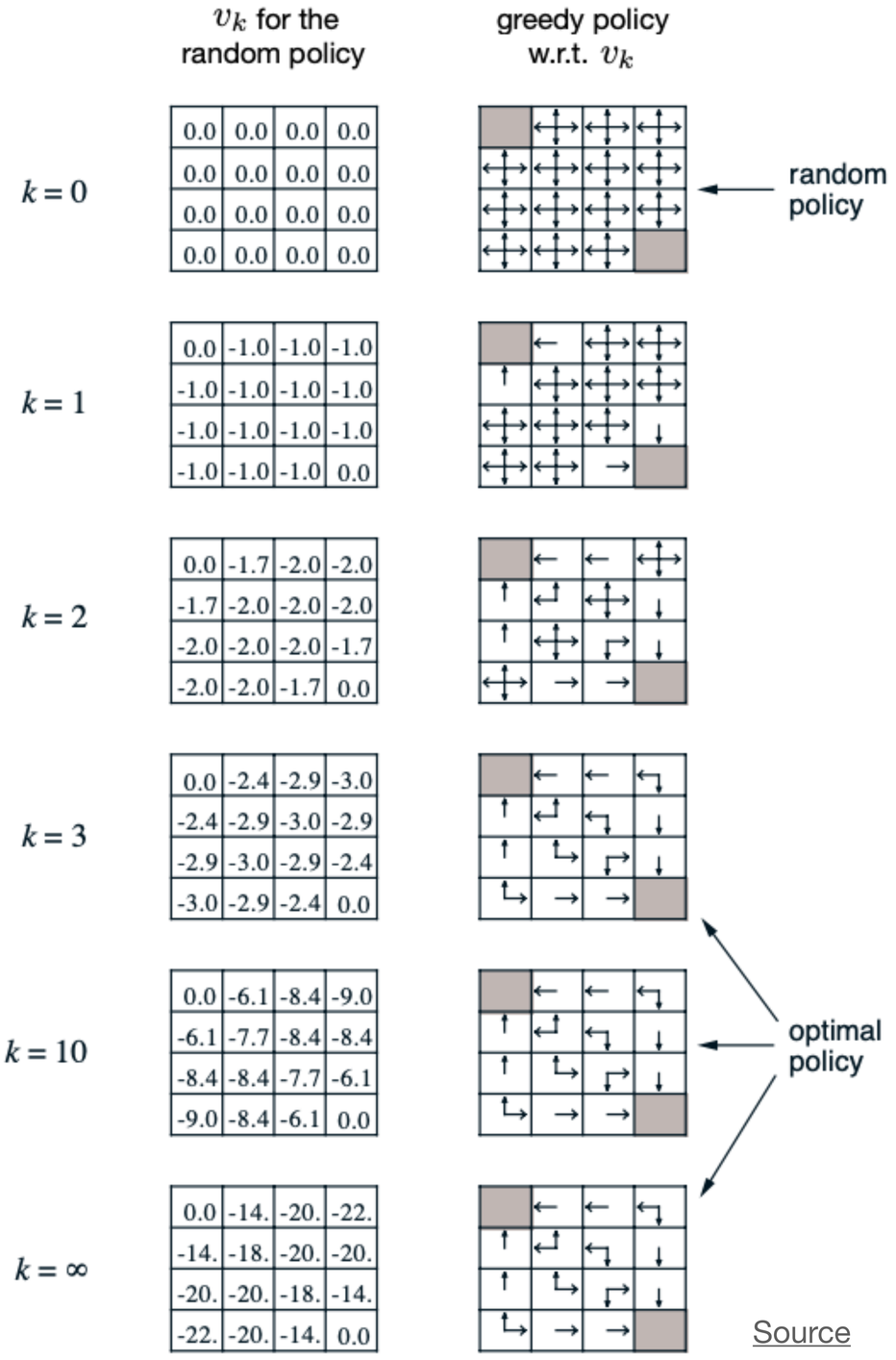
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
      $\Delta \leftarrow 0$
      For each $s \in \mathcal{S}$:
         $v \leftarrow V(s)$
         $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\left[r + \gamma V(s')\right]$
         $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

Bellman expectation operator for $V^{\pi}(s)$

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
      old-action $\leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
      If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

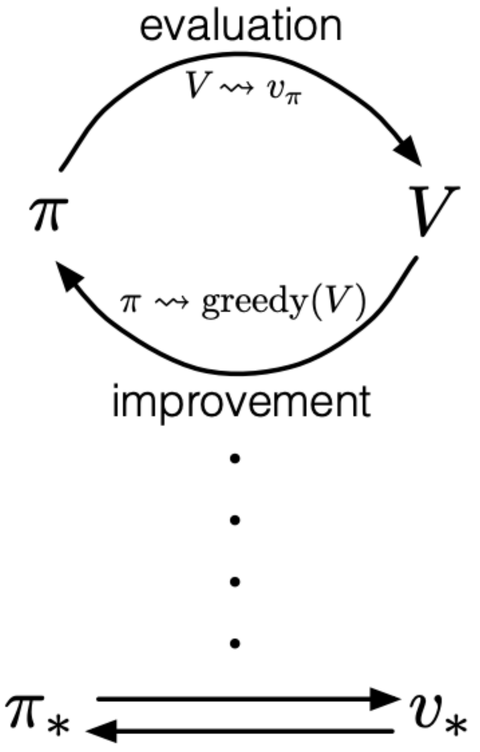
Almost Bellman optimality operator for $V^{\pi}(s)$


v_k for the random policy
greedy policy w.r.t. v_k
$k = 0$
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0
random policy
$k = 1$
0.0 -1.0 -1.0 -1.0
-1.0 -1.0 -1.0 -1.0
-1.0 -1.0 -1.0 -1.0
-1.0 -1.0 -1.0 0.0
$k = 2$
0.0 -1.7 -2.0 -2.0
-1.7 -2.0 -2.0 -2.0
-2.0 -2.0 -2.0 -1.7
-2.0 -2.0 -1.7 0.0
$k = 3$
0.0 -2.4 -2.9 -3.0
-2.4 -2.9 -3.0 -2.9
-2.9 -3.0 -2.9 -2.4
-3.0 -2.9 -2.4 0.0
$k = 10$
0.0 -6.1 -8.4 -9.0
-6.1 -7.7 -8.4 -8.4
-8.4 -8.4 -7.7 -6.1
-9.0 -8.4 -6.1 0.0
optimal policy
$k = \infty$
0.0 -14. -20. -22.
-14. -18. -20. -20.
-20. -20. -18. -14.
-22. -20. -14. 0.0

Source

Policy Iteration

Policy iteration consists of two processes, each completing before the other begins: one making the value function consistent with the current policy (policy evaluation), and the other making the policy greedy with respect to the current value function (policy improvement)



Source

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

Value Iteration

One drawback to policy iteration is that each of its iterations involves policy evaluation, which may itself be a protracted iterative computation requiring multiple sweeps through the state set. If policy evaluation is done iteratively, then convergence exactly to V^{π} occurs only in the limit.

Value Iteration

In fact, the policy evaluation step of policy iteration can be truncated in several ways without losing the convergence guarantees of policy iteration. One important special case is when policy evaluation is stopped after just one sweep (one update of each state). This algorithm is called value iteration.

$$V_{k+1}(s) = \max_a \sum_{r,s'} p(r, s' \mid s, a) \left[r + \gamma V_k(s') \right]$$

Value Iteration

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, $\pi \approx \pi_*$, such that
  $\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$

Synchronous Dynamic Programming Algorithms

Problem	Bellman Equation	Algorithm	Complexity per iteration:
Prediction	Bellman Expectation Equation	Iterative Policy Evaluation	
Control	Bellman Expectation Equation + (Greedy) Policy Improvement	Policy Iteration	$O(\|A\|\|S\|^2 + \|S\|^3)$
Control	Bellman Optimality Equation	Value Iteration	$O(\|A\|\|S\|^2)$

Source

Drawback: involve operations over the entire state set of the MDP, that is, they require sweeps of the state set.

Asynchronous Dynamic Programming

DP methods described so far used synchronous updates (all states in parallel)

Asynchronous DP:

- backs up states individually, in any order
- can significantly reduce computation
- guaranteed to converge if all states continue to be selected

Asynchronous Dynamic Programming

Three simple ideas for asynchronous dynamic programming:

1. In-place dynamic programming: do not copy V^{π} but make in-place update during iteration method;
2. Prioritised sweeping: select state based on magnitude of Bellman error
3. Real-time dynamic programming: consider only relevant states using samples $< s, a, r, s' >$

These ideas will be extremely useful in a model-free setting!

Thank you for your attention!