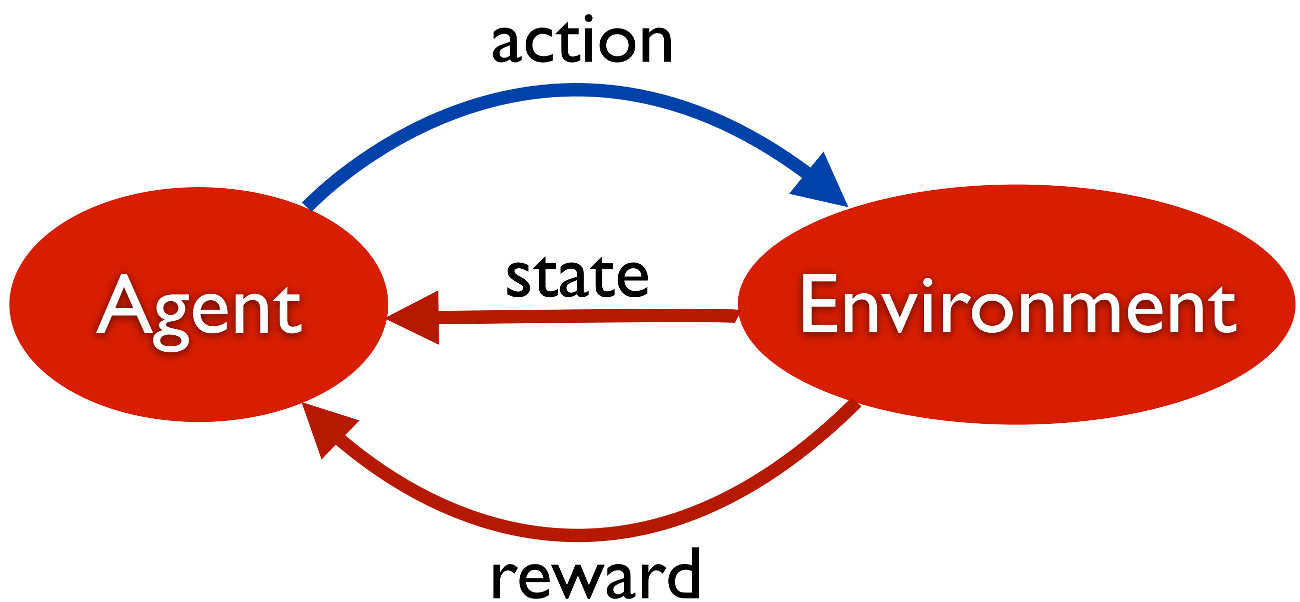
Foundations of Machine Learning
Reinforcement Learning

Mehryar Mohri
Courant Institute and Google Research
mohri@cims.nyu.edu

Reinforcement Learning

- Agent exploring environment.
- Interactions with environment:



- Problem: find action policy that maximizes cumulative reward over the course of interactions.

Key Features

- Contrast with supervised learning:
  - no explicit labeled training data.
  - distribution defined by actions taken.
- Delayed rewards or penalties.
- RL trade-off:
  - exploration (of unknown states and actions) to gain more reward information; vs.
  - exploitation (of known information) to optimize reward.

Applications

- Robot control e.g., Robocup Soccer Teams (Stone et al., 1999), helicopter flight, autonomous driving.
- Board games, e.g., TD-Gammon (Tesauro, 1995), Go (Silver et al., 2016).
- Elevator scheduling (Crites and Barto, 1996).
- Ads placement, patient treatment.
- Telecommunications.
- Inventory management.
- Dynamic radio channel assignment.

This Lecture

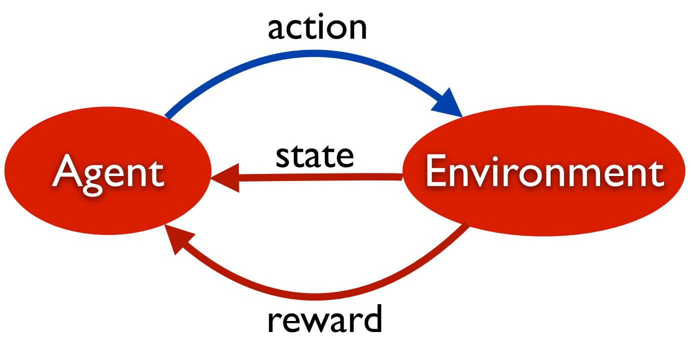
- Markov Decision Processes (MDPs)
- Planning
- Learning
- Multi-armed bandit problem

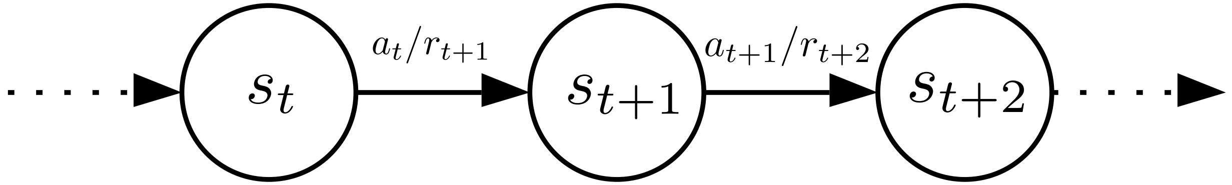
Markov Decision Process (MDP)

- Definition: a Markov Decision Process is defined by:
  - a set of decision epochs $\{0, \ldots, T\}$.
  - a set of states S, possibly infinite.
  - a start state or initial state s_0;
  - a set of actions A, possibly infinite.
  - a transition probability $\Pr[s'|s, a]$: distribution over destination states $s' = \delta(s, a)$.
  - a reward probability $\Pr[r'|s, a]$: distribution over rewards returned $r' = r(s, a)$.

Model

- State observed at time t : $s_t \in S$.
- Action taken at time t : $a_t \in A$.
- State reached $s_{t+1} = \delta(s_t, a_t)$.
- Reward received: $r_{t+1} = r(s_t, a_t)$.

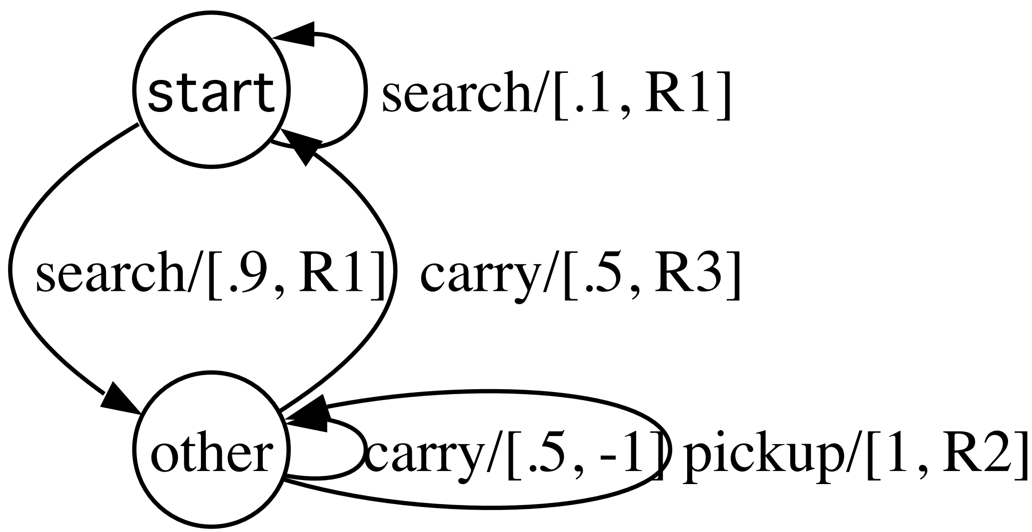




MDPs - Properties

- Finite MDPs: A and S finite sets.
- Finite horizon when $T < \infty$.
- Reward $r(s, a)$: often deterministic function.

Example - Robot Picking up Balls



Policy

- Definition: a policy is a mapping $\pi \colon S \to A$.
- Objective: find policy π maximizing expected return.
    - finite horizon return: $\sum_{t=0}^{T-1} r\big(s_t, \pi(s_t)\big)$.
    - infinite horizon return: $\sum_{t=0}^{+\infty} \gamma^t r\big(s_t, \pi(s_t)\big)$.
- Theorem: for any finite MDP, there exists an optimal policy (for any start state).

Policy Value

- **Definition**: the value of a policy π at state s is
  - finite horizon:

$$V_\pi(s) = \mathrm{E}\left[\sum_{t=0}^{T-1} r\big(s_t, \pi(s_t)\big) \,\Big|\, s_0 = s\right].$$

  - infinite horizon: discount factor $\gamma \in [0, 1)$,

$$V_\pi(s) = \mathrm{E}\left[\sum_{t=0}^{+\infty} \gamma^t r\big(s_t, \pi(s_t)\big) \,\Big|\, s_0 = s\right].$$

- **Problem**: find policy π with maximum value for all states.

Policy Evaluation

- Analysis of policy value:

$$\begin{aligned} V_\pi(s) &= \mathrm{E}\left[\sum_{t=0}^{+\infty} \gamma^t r\big(s_t, \pi(s_t)\big) \bigg| s_0 = s\right]. \\ &= \mathrm{E}[r(s, \pi(s))] + \gamma\, \mathrm{E}\left[\sum_{t=0}^{+\infty} \gamma^t r\big(s_{t+1}, \pi(s_{t+1})\big) \bigg| s_0 = s\right] \\ &= \mathrm{E}[r(s, \pi(s)] + \gamma\, \mathrm{E}[V_\pi(\delta(s, \pi(s)))]. \end{aligned}$$

- Bellman equations (system of linear equations):

$$\boxed{V_\pi(s) = \mathrm{E}[r(s, \pi(s)] + \gamma \sum_{s'} \Pr[s' | s, \pi(s)] V_\pi(s').}$$

Bellman Equation - Existence and Uniqueness

- Notation:
  - transition probability matrix $\mathbf{P}_{s,s'} = \Pr[s'|s, \pi(s)]$.
  - value column matrix $\mathbf{V} = V_\pi(s)$.
  - expected reward column matrix: $\mathbf{R} = \mathrm{E}[r(s, \pi(s)]$.
- Theorem: for a finite MDP, Bellman's equation admits a unique solution given by

$$\mathbf{V}_0 = (\mathbf{I} - \gamma \mathbf{P})^{-1} \mathbf{R}.$$

Bellman Equation - Existence and Uniqueness

- Proof: Bellman's equation rewritten as

$$\mathbf{V} = \mathbf{R} + \gamma \mathbf{P} \mathbf{V}.$$

  - $\mathbf{P}$ is a stochastic matrix, thus,

$$\|\mathbf{P}\|_\infty = \max_s \sum_{s'} |\mathbf{P}_{ss'}| = \max_s \sum_{s'} \Pr[s'|s, \pi(s)] = 1.$$

  - This implies that $\|\gamma \mathbf{P}\|_\infty = \gamma < 1.$ The eigenvalues of $\gamma \mathrm{P}$ are all less than one and $(\mathbf{I} - \gamma \mathbf{P})$ is invertible.

- Notes: general shortest distance problem (MM, 2002).

Optimal Policy

- Definition: policy π^* with maximal value for all states $s \in S$.
  - value of π^* (optimal value):
$$\forall s \in S, V_{\pi^*}(s) = \max_{\pi} V_\pi(s).$$
  - optimal state-action value function: expected return for taking action a at state s and then following optimal policy.
$$\begin{aligned} Q^*(s,a) &= \mathrm{E}[r(s,a)] + \gamma \, \mathrm{E}[V^*(\delta(s,a))] \\ &= \mathrm{E}[r(s,a)] + \gamma \sum_{s' \in S} \Pr[s' \mid s,a] V^*(s'). \end{aligned}$$

Optimal Values - Bellman Equations

- Property: the following equalities hold:

$$\forall s \in S, \;\; V^*(s) = \max_{a \in A} Q^*(s, a).$$

- Proof: by definition, for all s, $V^*(s) \leq \max_{a \in A} Q^*(s, a)$.
  - If for some s we had $V^*(s) < \max_{a \in A} Q^*(s, a)$, then maximizing action would define a better policy.

- Thus,

$$\boxed{V^*(s) = \max_{a \in A} \Big\{ \mathrm{E}[r(s, a)] + \gamma \sum_{s' \in S} \Pr[s' | s, a] V^*(s') \Big\}.}$$

This Lecture

- Markov Decision Processes (MDPs)
- Planning
- Learning
- Multi-armed bandit problem

Known Model

- Setting: environment model known.
- Problem: find optimal policy.
- Algorithms:
  - value iteration.
  - policy iteration.
  - linear programming.

Value Iteration Algorithm

$$\mathbf{\Phi}(\mathbf{V})(s) = \max_{a \in A} \Big\{ \mathrm{E}[r(s,a)] + \gamma \sum_{s' \in S} \Pr[s'|s,a] V(s') \Big\}.$$

$$\mathbf{\Phi}(\mathbf{V}) = \max_{\pi} \{ \mathbf{R}_\pi + \gamma \mathbf{P}_\pi \mathbf{V} \}.$$

ValueIteration($\mathbf{V}_0$)

1. $\mathbf{V} \leftarrow \mathbf{V}_0$ $\quad \triangleright$ $\mathbf{V}_0$ arbitrary value
2. **while** $\|\mathbf{V} - \mathbf{\Phi}(\mathbf{V})\| \geq \frac{(1-\gamma)\epsilon}{\gamma}$ **do**
3. $\quad\quad \mathbf{V} \leftarrow \mathbf{\Phi}(\mathbf{V})$
4. **return** $\mathbf{\Phi}(\mathbf{V})$

VI Algorithm - Convergence

- Theorem: for any initial value $\mathbf{V}_0$, the sequence defined by $\mathbf{V}_{n+1} = \mathbf{\Phi}(\mathbf{V}_n)$ converge to $\mathbf{V}^*$.
- Proof: we show that $\mathbf{\Phi}$ is γ-contracting for $\|\cdot\|_\infty$

→ existence and uniqueness of fixed point for $\mathbf{\Phi}$.

  - for any $s \in S$, let $a^*(s)$ be the maximizing action defining $\mathbf{\Phi}(\mathbf{V})(s)$. Then, for $s \in S$ and any $\mathbf{U}$,

$$
\begin{aligned}
\mathbf{\Phi}(\mathbf{V})(s) - \mathbf{\Phi}(\mathbf{U})(s) &\leq \mathbf{\Phi}(\mathbf{V})(s) - \Big(\mathrm{E}[r(s, a^*(s))] + \gamma \sum_{s' \in S} \Pr[s' \mid s, a^*(s)]\mathbf{U}(s')\Big) \\
&= \gamma \sum_{s' \in S} \Pr[s' | s, a^*(s)][\mathbf{V}(s') - \mathbf{U}(s')] \\
&\leq \gamma \sum_{s' \in S} \Pr[s' | s, a^*(s)] \|\mathbf{V} - \mathbf{U}\|_\infty = \gamma \|\mathbf{V} - \mathbf{U}\|_\infty.
\end{aligned}
$$

Complexity and Optimality

- Complexity: convergence in $O(\log \frac{1}{\epsilon})$. Observe that

$$\|\mathbf{V}_{n+1} - \mathbf{V}_n\|_\infty \leq \gamma \|\mathbf{V}_n - \mathbf{V}_{n-1}\|_\infty \leq \gamma^n \|\mathbf{\Phi}(\mathbf{V}_0) - \mathbf{V}_0\|_\infty.$$

Thus, $\gamma^n \|\mathbf{\Phi}(\mathbf{V}_0) - \mathbf{V}_0\|_\infty \leq \dfrac{(1-\gamma)\epsilon}{\gamma} \Rightarrow n = O\Big(\log \dfrac{1}{\epsilon}\Big).$

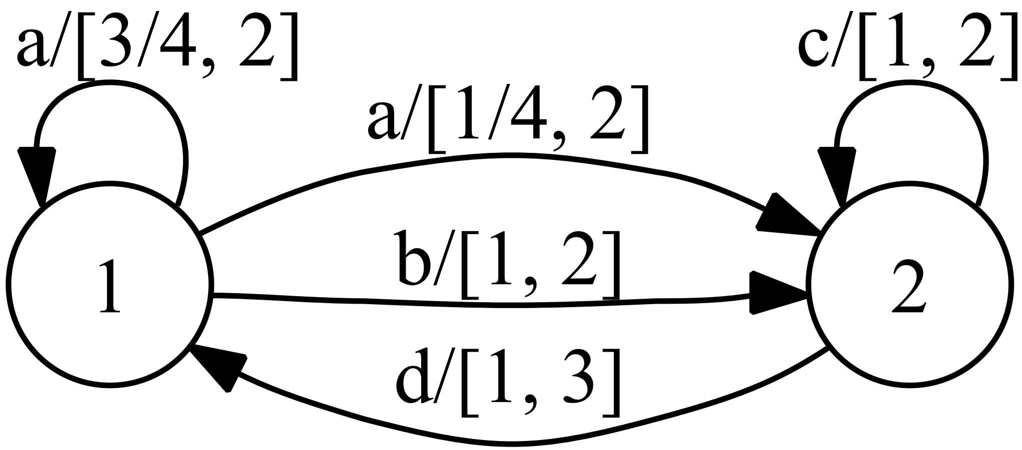
- ϵ-Optimality: let $\mathbf{V}_{n+1}$ be the value returned. Then,

$$\begin{aligned}\|\mathbf{V}^* - \mathbf{V}_{n+1}\|_\infty &\leq \|\mathbf{V}^* - \mathbf{\Phi}(\mathbf{V}_{n+1})\|_\infty + \|\mathbf{\Phi}(\mathbf{V}_{n+1}) - \mathbf{V}_{n+1}\|_\infty \\ &\leq \gamma \|\mathbf{V}^* - \mathbf{V}_{n+1}\|_\infty + \gamma \|\mathbf{V}_{n+1} - \mathbf{V}_n\|_\infty.\end{aligned}$$

Thus,

$$\|\mathbf{V}^* - \mathbf{V}_{n+1}\|_\infty \leq \frac{\gamma}{1-\gamma} \|\mathbf{V}_{n+1} - \mathbf{V}_n\|_\infty \leq \epsilon.$$

VI Algorithm - Example



$$\mathbf{V}_{n+1}(1) = \max\left\{2 + \gamma\Big(\frac{3}{4}\mathbf{V}_n(1) + \frac{1}{4}\mathbf{V}_n(2)\Big), 2 + \gamma\mathbf{V}_n(2)\right\}$$

$$\mathbf{V}_{n+1}(2) = \max\left\{3 + \gamma\mathbf{V}_n(1), 2 + \gamma\mathbf{V}_n(2)\right\}.$$

For $\mathbf{V}_0(1) = -1, \mathbf{V}_0(2) = 1, \gamma = 1/2$, $\mathbf{V}_1(1) = \mathbf{V}_1(2) = 5/2.$

But, $\mathbf{V}^*(1) = 14/3, \mathbf{V}^*(2) = 16/3.$

Policy Iteration Algorithm

PolicyIteration(π_0)

$\pi \leftarrow \pi_0 \quad \triangleright\, \pi_0$ arbitrary policy
$\pi' \leftarrow$ NIL
while $(\pi \neq \pi')$ **do**
  $\mathbf{V} \leftarrow \mathbf{V}_\pi \quad \triangleright$ policy evaluation: solve $(\mathbf{I} - \gamma \mathbf{P}_\pi)\mathbf{V} = \mathbf{R}_\pi$.
  $\pi' \leftarrow \pi$
  $\pi \leftarrow \operatorname{argmax}_\pi \{\mathbf{R}_\pi + \gamma \mathbf{P}_\pi \mathbf{V}\} \quad \triangleright$ greedy policy improvement.
return π

PI Algorithm - Convergence

- Theorem: let $(\mathbf{V}_n)_{n\in\mathbb{N}}$ be the sequence of policy values computed by the algorithm, then,

$$\mathbf{V}_n \leq \mathbf{V}_{n+1} \leq \mathbf{V}^*.$$

- Proof: let π_{n+1} be the policy improvement at the nth iteration, then, by definition,

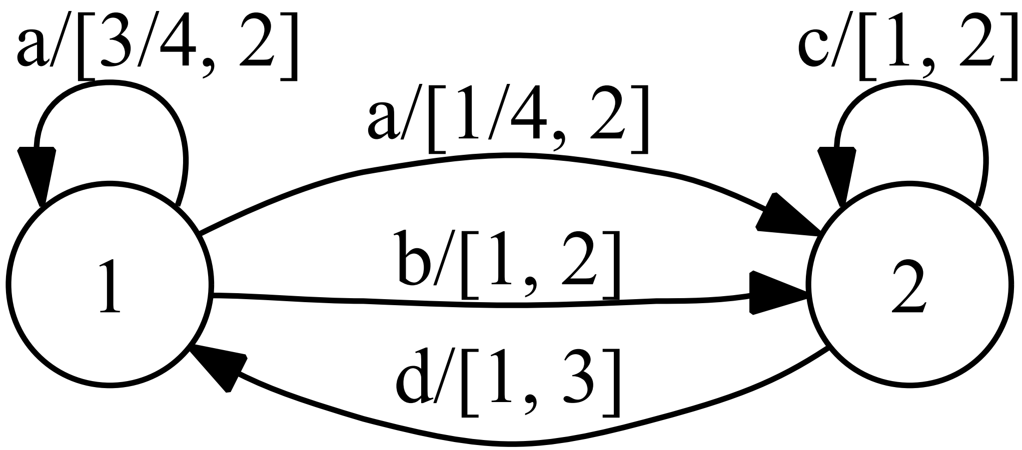
$$\mathbf{R}_{\pi_{n+1}} + \gamma \mathbf{P}_{\pi_{n+1}} \mathbf{V}_n \geq \mathbf{R}_{\pi_n} + \gamma \mathbf{P}_{\pi_n} \mathbf{V}_n = \mathbf{V}_n.$$

  - therefore, $\mathbf{R}_{\pi_{n+1}} \geq (\mathbf{I} - \gamma \mathbf{P}_{\pi_{n+1}}) \mathbf{V}_n$.
  - note that $(\mathbf{I} - \gamma \mathbf{P}_{\pi_{n+1}})^{-1}$ preserves ordering:
    $\mathbf{X} \geq \mathbf{0} \Rightarrow (\mathbf{I} - \gamma \mathbf{P}_{\pi_{n+1}})^{-1} \mathbf{X} = \sum_{k=0}^{\infty} (\gamma \mathbf{P}_{\pi_{n+1}})^k \mathbf{X} \geq \mathbf{0}$.
  - thus, $\mathbf{V}_{n+1} = (\mathbf{I} - \gamma \mathbf{P}_{\pi_{n+1}})^{-1} \mathbf{R}_{\pi_{n+1}} \geq \mathbf{V}_n$.

Notes

- Two consecutive policy values can be equal only at last iteration.
- The total number of possible policies is $|A|^{|S|}$, thus, this is the maximal possible number of iterations.
  - best upper bound known $O\left(\frac{|A|^{|S|}}{|S|}\right)$.

PI Algorithm - Example



Initial policy: $\pi_0(1) = b, \pi_0(2) = c.$

Evaluation: $V_{\pi_0}(1) = 1 + \gamma V_{\pi_0}(2)$

$$V_{\pi_0}(2) = 2 + \gamma V_{\pi_0}(2).$$

Thus, $V_{\pi_0}(1) = \dfrac{1+\gamma}{1-\gamma} \quad V_{\pi_0}(2) = \dfrac{2}{1-\gamma}.$

VI and PI Algorithms - Comparison

- Theorem: let $(\mathbf{U}_n)_{n\in\mathbb{N}}$ be the sequence of policy values generated by the VI algorithm, and $(\mathbf{V}_n)_{n\in\mathbb{N}}$ the one generated by the PI algorithm. If $\mathbf{U}_0 = \mathbf{V}_0$, then,

$$\forall n \in \mathbb{N},\ \mathbf{U}_n \leq \mathbf{V}_n \leq \mathbf{V}^*.$$

- Proof: we first show that $\mathbf{\Phi}$ is monotonic. Let $\mathbf{U}$ and $\mathbf{V}$ be such that $\mathbf{U} \leq \mathbf{V}$ and let π be the policy such that $\mathbf{\Phi}(\mathbf{U}) = \mathbf{R}_\pi + \gamma \mathbf{P}_\pi \mathbf{U}$. Then,

$$\mathbf{\Phi}(\mathbf{U}) \leq \mathbf{R}_\pi + \gamma \mathbf{P}_\pi \mathbf{V} \leq \max_{\pi'}\{\mathbf{R}'_\pi + \gamma \mathbf{P}'_\pi \mathbf{V}\} = \mathbf{\Phi}(\mathbf{V}).$$

VI and PI Algorithms - Comparison

- The proof is by induction on n. Assume $\mathbf{U}_n \leq \mathbf{V}_n$, then, by the monotonicity of $\mathbf{\Phi}$,

$$\mathbf{U}_{n+1} = \mathbf{\Phi}(\mathbf{U}_n) \leq \mathbf{\Phi}(\mathbf{V}_n) = \max_{\pi} \{\mathbf{R}_\pi + \gamma \mathbf{P}_\pi \mathbf{V}_n\}.$$

- Let π_{n+1} be the maximizing policy:

$$\pi_{n+1} = \operatorname*{argmax}_{\pi} \{\mathbf{R}_\pi + \gamma \mathbf{P}_\pi \mathbf{V}_n\}.$$

- Then,

$$\mathbf{\Phi}(\mathbf{V}_n) = \mathbf{R}_{\pi_{n+1}} + \gamma \mathbf{P}_{\pi_{n+1}} \mathbf{V}_n \leq \mathbf{R}_{\pi_{n+1}} + \gamma \mathbf{P}_{\pi_{n+1}} \mathbf{V}_{n+1} = \mathbf{V}_{n+1}.$$

Notes

- The PI algorithm converges in a smaller number of iterations than the VI algorithm due to the optimal policy.
- But, each iteration of the PI algorithm requires computing a policy value, i.e., solving a system of linear equations, which is more expensive to compute that an iteration of the VI algorithm.

Primal Linear Program

- LP formulation: choose $\alpha(s) > 0$, with $\sum_s \alpha(s) = 1$.

$$\min_{\mathbf{V}} \sum_{s \in S} \alpha(s) V(s)$$

$$\text{subject to} \;\; \forall s \in S, \forall a \in A, V(s) \geq \mathrm{E}[r(s,a)] + \gamma \sum_{s' \in S} \Pr[s'|s,a] V(s').$$

- Parameters:
  - number rows: $|S||A|$.
  - number of columns: $|S|$.

Dual Linear Program

- LP formulation:

$$\max_{\mathbf{x}} \sum_{s \in S, a \in A} \mathrm{E}[r(s,a)]\, x(s,a)$$

$$\text{subject to} \quad \forall s \in S, \sum_{a \in A} x(s',a) = \alpha(s') + \gamma \sum_{s \in S, a \in A} \Pr[s'|s,a]\, x(s',a)$$

$$\forall s \in S, \forall a \in A, x(s,a) \geq 0.$$

- Parameters: more favorable number of rows.
  - number rows: $|S|$.
  - number of columns: $|S||A|$.

This Lecture

- Markov Decision Processes (MDPs)
- Planning
- Learning
- Multi-armed bandit problem

Problem

- Unknown model:
  - transition and reward probabilities not known.
  - realistic scenario in many practical problems, e.g., robot control.
- Training information: sequence of immediate rewards based on actions taken.
- Learning approches:
  - model-free: learn policy directly.
  - model-based: learn model, use it to learn policy.

Learning Approaches

- Two broad families:
  - model-based approaches: use samples based on interactions to learn P and r explicitly; next, use value iteration to learn policy.
  - model-free approaches: do not seek to learn model; instead, use samples to learn Q function; policy readily derived from Q.

Problem

- How do we estimate reward and transition probabilities?
  - use equations derived for policy value and Q-functions.
  - but, equations given in terms of some expectations.
  - ⟶ instance of a stochastic approximation problem.

Stochastic Approximation

- Problem: find solution of $\mathbf{x} = H(\mathbf{x})$ with $\mathbf{x} \in \mathbb{R}^N$ while
  - $H(\mathbf{x})$ cannot be computed, e.g., H not accessible;
  - i.i.d. sample of noisy observations $H(\mathbf{x}_i) + \mathbf{w}_i$, available, $i \in [1, m]$, with $\mathrm{E}[\mathbf{w}] = 0$.
- Idea: algorithm based on iterative technique:

$$\begin{aligned} \mathbf{x}_{t+1} &= (1 - \alpha_t)\mathbf{x}_t + \alpha_t[H(\mathbf{x}_t) + \mathbf{w}_t] \\ &= \mathbf{x}_t + \alpha_t[H(\mathbf{x}_t) + \mathbf{w}_t - \mathbf{x}_t]. \end{aligned}$$

  - more generally $\mathbf{x}_{t+1} = \mathbf{x}_t + \alpha_t D(\mathbf{x}_t, \mathbf{w}_t)$.

Mean Estimation

- Theorem: Let X be a random variable taking values in $[0, 1]$ and let $x_0, \ldots, x_m$ be i.i.d. values of X. Define the sequence $(\mu_m)_{m \in \mathbb{N}}$ by

$$\mu_{m+1} = (1 - \alpha_m)\mu_m + \alpha_m x_m \quad \text{with } \mu_0 = x_0.$$

Then, for $\alpha_m \in [0, 1]$, with $\sum_{m \geq 0} \alpha_m = +\infty$ and $\sum_{m \geq 0} \alpha_m^2 < +\infty$,

$$\boxed{\mu_m \xrightarrow{\text{a.s}} \mathrm{E}[X].}$$

Proof

- Proof: By the independence assumption, for $m \geq 0$,

$$\begin{aligned}\mathrm{Var}[\mu_{m+1}] &= (1-\alpha_m)^2 \mathrm{Var}[\mu_m] + \alpha_m^2 \mathrm{Var}[x_m] \\ &\leq (1-\alpha_m)\mathrm{Var}[\mu_m] + \alpha_m^2.\end{aligned}$$

  - We have $\alpha_m \to 0$ since $\sum_{m \geq 0} \alpha_m^2 < +\infty$.
  - Let $\epsilon > 0$ and suppose there exists $N \in \mathbb{N}$ such that for all $m \geq N$, $\mathrm{Var}[\mu_m] \geq \epsilon$. Then, for $m \geq N$,

$$\mathrm{Var}[\mu_{m+1}] \leq \mathrm{Var}[\mu_m] - \alpha_m \epsilon + \alpha_m^2,$$

which implies $\mathrm{Var}[\mu_{m+N}] \leq \underbrace{\mathrm{Var}[\mu_N] - \epsilon \sum_{n=N}^{m+N} \alpha_n + \sum_{n=N}^{m+N} \alpha_n^2}_{\to -\infty \text{ when } m \to \infty},$

contradicting $\mathrm{Var}[\mu_{m+N}] \geq 0$.

Mean Estimation

- Thus, for all $N \in \mathbb{N}$ there exists $m_0 \geq N$ such that $\mathrm{Var}[\mu_{m_0}] < \epsilon$. Choose N large enough so that $\forall m \geq N, \alpha_m \leq \epsilon$. Then,

$$\mathrm{Var}[\mu_{m_0+1}] \leq (1 - \alpha_{m_0})\epsilon + \epsilon \alpha_{m_0} = \epsilon.$$

- Therefore, $\mu_m \leq \epsilon$ for all $m \geq m_0$ (L_2 convergence).

Notes

- special case: $\alpha_m = \frac{1}{m}$.
  - Strong law of large numbers.
- Connection with stochastic approximation.

TD(0) Algorithm

- Idea: recall Bellman's linear equations giving V

$$V_\pi(s) = \mathrm{E}[r(s, \pi(s)] + \gamma \sum_{s'} \Pr[s'|s, \pi(s)] V_\pi(s')$$
$$= \mathop{\mathrm{E}}_{s'}\left[r(s, \pi(s)) + \gamma V_\pi(s')|s\right].$$

- Algorithm: temporal difference (TD).
  - sample new state s'.
  - update: α depends on number of visits of s.

$$V(s) \leftarrow (1-\alpha)V(s) + \alpha[r(s, \pi(s)) + \gamma V(s')]$$
$$= V(s) + \alpha[\underbrace{r(s, \pi(s)) + \gamma V(s') - V(s)}_{\text{temporal difference of } V \text{ values}}].$$

TD(0) Algorithm

TD(0)()

V $\leftarrow$ $\mathbf{V}_0$ $\triangleright$ initialization.
for $t \leftarrow 0$ **to** T **do**
  $s \leftarrow$ SelectState()
  for each step of epoch t **do**
    $r' \leftarrow$ Reward$(s, \pi(s))$
    $s' \leftarrow$ NextState(π, s)
    $V(s) \leftarrow (1-\alpha)V(s) + \alpha[r' + \gamma V(s')]$
    $s \leftarrow s'$
return V

Q-Learning Algorithm

- Idea: assume deterministic rewards.

$$Q^*(s,a) = \mathrm{E}[r(s,a)] + \gamma \sum_{s' \in S} \Pr[s' \mid s,a] V^*(s')$$
$$= \mathop{\mathrm{E}}_{s'}[r(s,a) + \gamma \max_{a \in A} Q^*(s',a)]$$

- Algorithm: $\alpha \in [0,1]$ depends on number of visits.
  - sample new state s'.
  - update:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[r(s,a) + \gamma \max_{a' \in A} Q(s',a')].$$

Q-Learning Algorithm

(Watkins, 1989; Watkins and Dayan 1992)

Q-Learning(π)

$Q \leftarrow Q_0$ $\triangleright$ initialization, e.g., $Q_0 = 0$.
for $t \leftarrow 0$ **to** T **do**
  $s \leftarrow$ SelectState()
  for each step of epoch t **do**
    $a \leftarrow$ SelectAction(π, s) $\triangleright$ policy π derived from Q, e.g., ϵ-greedy.
    $r' \leftarrow$ Reward(s, a)
    $s' \leftarrow$ NextState(s, a)
    $Q(s,a) \leftarrow Q(s,a) + \alpha\big[r' + \gamma \max_{a'} Q(s',a') - Q(s,a)\big]$
    $s \leftarrow s'$
return Q

Notes

- Can be viewed as a stochastic formulation of the value iteration algorithm.
- Convergence for any policy so long as states and actions visited infinitely often and parameter chosen as in mean estimation theorem.
- How to choose the action at each iteration? Maximize reward? Explore other actions?
- Q-learning is an off-policy method: no control over the policy; estimates and evaluates policy using experience from following different policy.

Policies

- Epsilon-greedy strategy:
  - with probability $1-\epsilon$ greedy action from s ;
  - with probability ϵ random action.
- Epoch-dependent strategy (Boltzmann exploration):

$$p_t(a|s, Q) = \frac{e^{\frac{Q(s,a)}{\tau_t}}}{\sum_{a' \in A} e^{\frac{Q(s,a')}{\tau_t}}},$$

  - $\tau_t \rightarrow 0$: greedy selection.
  - larger τ_t : random action.

Convergence of Q-Learning

- Theorem: consider a finite MDP. Assume that for all $s \in S$ and $a \in A$, $\sum_{t=0}^{\infty} \alpha_t(s, a) = \infty, \sum_{t=0}^{\infty} \alpha_t^2(s, a) < \infty$ with $\alpha_t(s, a) \in [0, 1]$. Then, the Q-learning algorithm converges to the optimal value Q^* (with probability one).
  - note: the conditions on $\alpha_t(s, a)$ impose that each state-action pair is visited infinitely many times.

This Lecture

- Markov Decision Processes (MDPs)
- Planning
- Learning
- Multi-armed bandit problem

Multi-Armed Bandit Problem

(Robbins, 1952)

- Problem: gambler must decide which arm of a N-slot machine to pull to maximize his total reward in a series of trials.
  - stochastic setting: N lever reward distributions.
  - adversarial setting: reward selected by adversary aware of all the past.

Applications

- Clinical trials.
- Adaptive routing.
- Ads placement on pages.
- Games.

Multi-Armed Bandit Game

- For $t = 1$ to T do
  - adversary determines outcome $y_t \in Y$.
  - player selects probability distribution p_t and pulls lever $I_t \in \{1, \ldots, N\}$, $I_t \sim p_t$.
  - player incurs loss $L(I_t, y_t)$ (adversary is informed of p_t and I_t.
- Objective: minimize regret

$$\text{Regret}(T) = \sum_{t=1}^{T} L(I_t, y_t) - \min_{i=1,\ldots,N} \sum_{t=1}^{T} L(i, y_t).$$

Notes

- Player is informed only of the loss (or reward) corresponding to his own action.
- Adversary knows past but not action selected.
- Stochastic setting: loss $(L(1, y_t), \ldots, L(N, y_t))$ drawn according to some distribution $D = D_1 \otimes \cdots \otimes D_N$. Regret definition modified by taking expectations.
- Exploration/Exploitation trade-off: playing the best arm found so far versus seeking to find an arm with a better payoff.

Notes

- Equivalent views:
  - special case of learning with partial information.
  - one-state MDP learning problem.
- Simple strategy: ϵ-greedy: play arm with best empirical reward with probability $1-\epsilon_t$, random arm with probability ϵ_t.

Exponentially Weighted Average

- Algorithm: Exp3, defined for $\eta, \gamma > 0$ by

$$p_{i,t} = (1-\gamma)\frac{\exp\big(-\eta\sum_{s=1}^{t-1}\widehat{l}_{i,t}\big)}{\sum_{i=1}^{N}\exp\big(-\eta\sum_{s=1}^{t-1}\widehat{l}_{i,t}\big)} + \frac{\gamma}{N},$$

with $\forall i \in [1, N], \widehat{l}_{i,t} = \frac{L(I_t, y_t)}{p_{I_t,t}} 1_{I_t = i}$.

- Guarantee: expected regret of

$$O(\sqrt{NT\log N}).$$

Exponentially Weighted Average

- Proof: similar to the one for the Exponentially Weighted Average with the additional observation that:

$$\mathrm{E}[\widehat{l}_{i,t}] = \sum_{i=1}^{N} p_{i,t} \frac{L(I_t, y_t)}{p_{I_t,t}} 1_{I_t=i} = L(i, y_t).$$

References

- Dimitri P. Bertsekas. Dynamic Programming and Optimal Control. 2 vols. Belmont, MA: Athena Scientific, 2007.
- Mehryar Mohri. Semiring Frameworks and Algorithms for Shortest-Distance Problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321-350, 2002.
- Martin L. Puterman *Markov decision processes: discrete stochastic dynamic programming*. Wiley-Interscience, New York, 1994.
- Robbins, H. (1952), "Some aspects of the sequential design of experiments", Bulletin of the American Mathematical Society 58 (5): 527–535.
- Sutton, Richard S., and Barto, Andrew G. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

References

- Gerald Tesauro. *Temporal Difference Learning and TD-Gammon*. Communications of the ACM 38 (3), 1995.

- Watkins, Christopher J. C. H. *Learning from Delayed Rewards*. Ph.D. thesis, Cambridge University, 1989.

- Christopher J. C. H. Watkins and Peter Dayan. *Q-learning*. Machine Learning, Vol. 8, No. 3-4, 1992.