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:
  - action
  - state
  - reward

- **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).
- Board games, e.g., TD-Gammon (Tesauro, 1995).
- Elevator scheduling (Crites and Barto, 1996).
- Ads placement.
- 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

- **start**
  - search/[0.1, R1]
- **search/[0.9, R1]**
- **other**
  - carry/[0.5, R3]
  - carry/[0.5, -1]
  - pickup/[1, R2]
Policy

Definition: a policy is a mapping $\pi: S \rightarrow A$.

Objective: find policy $\pi$ maximizing expected return.
- finite horizon return: $\sum_{t=0}^{T-1} r(s_t, \pi(s_t))$.
- infinite horizon return: $\sum_{t=0}^{+\infty} \gamma^t r(s_t, \pi(s_t))$.

Theorem: for any finite MDP, there exists an optimal policy (for any start state).
Policy Value

Definition: the value of a policy $\pi$ at state $s$ is

- **finite horizon:**
  \[ V_\pi(s) = \mathbb{E} \left[ \sum_{t=0}^{T-1} r(s_t, \pi(s_t)) \middle| s_0 = s \right] . \]

- **infinite horizon:** discount factor $\gamma \in [0, 1)$,
  \[ V_\pi(s) = \mathbb{E} \left[ \sum_{t=0}^{+\infty} \gamma^t r(s_t, \pi(s_t)) \middle| s_0 = s \right] . \]

Problem: find policy $\pi$ with maximum value for all states.
Policy Evaluation

- **Analysis of policy value:**

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

  \[ = E[r(s, \pi(s)) + \gamma E \left[ \sum_{t=0}^{+\infty} \gamma^t r(s_{t+1}, \pi(s_{t+1})) \right] | s_0 = s] \]

  \[ = E[r(s, \pi(s)) + \gamma E[V_{\pi}(\delta(s, \pi(s)))]]. \]

- **Bellman equations** (system of linear equations):

  \[ V_{\pi}(s) = E[r(s, \pi(s)) + \gamma \sum_{s'} \Pr[s' | s, \pi(s)] V_{\pi}(s')]. \]
Bellman Equation - Existence and Uniqueness

- **Notation:**
  - transition probability matrix $P_{s,s'} = \Pr[s'|s, \pi(s)]$.
  - value column matrix $V = V_\pi(s)$.
  - expected reward column matrix: $R = E[r(s, \pi(s)]$.

- **Theorem:** for a finite MDP, Bellman’s equation admits a unique solution given by

  $$V_0 = (I - \gamma P)^{-1} R.$$
Bellman Equation - Existence and Uniqueness

**Proof:** Bellman’s equation rewritten as

\[ V = R + \gamma PV. \]

- \( P \) is a stochastic matrix, thus,

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

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

**Notes:** general shortest distance problem (MM, 2002).
Optimal Policy

**Definition:** policy $\pi^*$ with maximal value for all states $s \in S$.

- **value of $\pi^*$ (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.
  $$Q^*(s, a) = \mathbb{E}[r(s, a)] + \gamma \mathbb{E}[V^*(\delta(s, a))]$$
  $$= \mathbb{E}[r(s, a)] + \gamma \sum_{s' \in S} \Pr[s' \mid s, a] V^*(s').$$
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,

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

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

\[ \Phi(V) = \max_{\pi} \{ R_\pi + \gamma P_\pi V \}. \]

**VALUE ITERATION**\((V_0)\)

1. \( V \leftarrow V_0 \quad \triangleright \text{ } V_0 \text{ arbitrary value} \\
2. \text{while } \|V - \Phi(V)\| \geq \frac{(1-\gamma)\epsilon}{\gamma} \text{ do} \\
3. \quad V \leftarrow \Phi(V) \\
4. \text{return } \Phi(V) \]
VI Algorithm - Convergence

Theorem: for any initial value $V_0$, the sequence defined by $V_{n+1} = \Phi(V_n)$ converge to $V^*$.

Proof: we show that $\Phi$ is $\gamma$-contracting for $\| \cdot \|_\infty$ existence and uniqueness of fixed point for $\Phi$.

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

$$
\Phi(V)(s) - \Phi(U)(s) \leq \Phi(V)(s) - \left( \mathbb{E}[r(s, a^*(s))] + \gamma \sum_{s' \in S} \Pr[s' \mid s, a^*(s)] U(s') \right)
$$

$$
= \gamma \sum_{s' \in S} \Pr[s' \mid s, a^*(s)] [V(s') - U(s')]
$$

$$
\leq \gamma \sum_{s' \in S} \Pr[s' \mid s, a^*(s)] \| V - U \|_\infty = \gamma \| V - U \|_\infty.
$$
Complexity and Optimality

- **Complexity**: convergence in \(O(\log \frac{1}{\epsilon})\). Observe that
\[
\|V_{n+1} - V_n\|_{\infty} \leq \gamma \|V_n - V_{n-1}\|_{\infty} \leq \gamma^n \|\Phi(V_0) - V_0\|_{\infty}.
\]
Thus, \(\gamma^n \|\Phi(V_0) - V_0\|_{\infty} \leq \frac{(1 - \gamma)\epsilon}{\gamma} \Rightarrow n = O\left(\log \frac{1}{\epsilon}\right)\).

- **\(\epsilon\)-Optimality**: let \(V_{n+1}\) be the value returned. Then,
\[
\|V^* - V_{n+1}\|_{\infty} \leq \|V^* - \Phi(V_{n+1})\|_{\infty} + \|\Phi(V_{n+1}) - V_{n+1}\|_{\infty}
\]
\[
\leq \gamma \|V^* - V_{n+1}\|_{\infty} + \gamma \|V_{n+1} - V_n\|_{\infty}.
\]
Thus,
\[
\|V^* - V_{n+1}\|_{\infty} \leq \frac{\gamma}{1 - \gamma} \|V_{n+1} - V_n\|_{\infty} \leq \epsilon.
\]
VI Algorithm - Example

\[ V_{n+1}(1) = \max \left\{ 2 + \gamma \left( \frac{3}{4} V_n(1) + \frac{1}{4} V_n(2) \right), 2 + \gamma V_n(2) \right\} \]

\[ V_{n+1}(2) = \max \left\{ 3 + \gamma V_n(1), 2 + \gamma V_n(2) \right\}. \]

For \( V_0(1) = -1, V_0(2) = 1, \gamma = 1/2, V_1(1) = V_1(2) = 5/2. \)

But, \( V^*(1) = 14/3, V^*(2) = 16/3. \)
Policy Iteration Algorithm

$\text{PolicyIteration}(\pi_0)$

1. $\pi \leftarrow \pi_0$ \(\triangleright\) $\pi_0$ arbitrary policy
2. $\pi' \leftarrow \text{NIL}$
3. while ($\pi \neq \pi'$) do
4. \hspace{1em} $V \leftarrow V_\pi$ \(\triangleright\) policy evaluation: solve $(I - \gamma P_\pi)V = R_\pi$.
5. \hspace{1em} $\pi' \leftarrow \pi$
6. \hspace{1em} $\pi \leftarrow \arg\max_\pi \{R_\pi + \gamma P_\pi V\}$ \(\triangleright\) greedy policy improvement.
7. return $\pi$
Theorem: let \((V_n)_{n \in \mathbb{N}}\) be the sequence of policy values computed by the algorithm, then,
\[ V_n \leq V_{n+1} \leq V^*. \]

Proof: let \(\pi_{n+1}\) be the policy improvement at the \(n\)th iteration, then, by definition,
\[ R_{\pi_{n+1}} + \gamma P_{\pi_{n+1}} V_n \geq R_{\pi_n} + \gamma P_{\pi_n} V_n = V_n. \]

- therefore, \(R_{\pi_{n+1}} \geq (I - \gamma P_{\pi_{n+1}}) V_n.\)
- note that \((I - \gamma P_{\pi_{n+1}})^{-1}\) preserves ordering:
\[ X \geq 0 \Rightarrow (I - \gamma P_{\pi_{n+1}})^{-1} X = \sum_{k=0}^{\infty} (\gamma P_{\pi_{n+1}})^k X \geq 0. \]
- thus, \(V_{n+1} = (I - \gamma P_{\pi_{n+1}})^{-1} R_{\pi_{n+1}} \geq V_n.\)
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 $O\left(\frac{|A|^{|S|}}{|S|}\right)$. 
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, $V_{\pi_0}(1) = \frac{1 + \gamma}{1 - \gamma}$, $V_{\pi_0}(2) = \frac{2}{1 - \gamma}$. 
VI and PI Algorithms - Comparison

**Theorem:** let \((U_n)_{n \in \mathbb{N}}\) be the sequence of policy values generated by the VI algorithm, and \((V_n)_{n \in \mathbb{N}}\) the one generated by the PI algorithm. If \(U_0 = V_0\), then,

\[\forall n \in \mathbb{N}, \ U_n \leq V_n \leq V^*.\]

**Proof:** we first show that \(\Phi\) is monotonic. Let \(U\) and \(V\) be such that \(U \leq V\) and let \(\pi\) be the policy such that \(\Phi(U) = R_\pi + \gamma P_\pi U\). Then,

\[\Phi(U) \leq R_\pi + \gamma P_\pi V \leq \max_{\pi'} \left\{ R'_{\pi} + \gamma P'_{\pi} V \right\} = \Phi(V).\]
VI and PI Algorithms - Comparison

- The proof is by induction on $n$. Assume $U_n \leq V_n$, then, by the monotonicity of $\Phi$,

$$U_{n+1} = \Phi(U_n) \leq \Phi(V_n) = \max_\pi \{R_\pi + \gamma P_\pi V_n\}.$$

- Let $\pi_{n+1}$ be the maximizing policy:

$$\pi_{n+1} = \arg\max_\pi \{R_\pi + \gamma P_\pi V_n\}.$$

- Then,

$$\Phi(V_n) = R_{\pi_{n+1}} + \gamma P_{\pi_{n+1}} V_n \leq R_{\pi_{n+1}} + \gamma P_{\pi_{n+1}} V_{n+1} = V_{n+1}.$$
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 than an iteration of the VI algorithm.
Primal Linear Program

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

\[
\min_V \sum_{s \in S} \alpha(s)V(s)
\]

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

**Parameters:**

- **number rows:** \(|S||A|\).
- **number of columns:** \(|S|\).
Dual Linear Program

- **LP formulation:**

\[
\max_x \sum_{s \in S, a \in A} E[r(s, a)] x(s, a)
\]

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

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

- **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 approaches:
- model-free: learn policy directly.
- model-based: learn model, use it to learn policy.
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 $x = H(x)$ with $x \in \mathbb{R}^N$ while
  - $H(x)$ cannot be computed, e.g., $H$ not accessible;
  - i.i.d. sample of noisy observations $H(x_i) + w_i$, available, $i \in [1, m]$, with $\mathbb{E}[w] = 0$.

- **Idea**: algorithm based on iterative technique:
  $$
x_{t+1} = (1 - \alpha_t)x_t + \alpha_t[H(x_t) + w_t] 
= x_t + \alpha_t[H(x_t) + w_t - x_t].
$$
  
  - more generally $x_{t+1} = x_t + \alpha_t D(x_t, w_t)$. 

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

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

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

$$\mu_m \xrightarrow{a.s.} \mathbb{E}[X].$$
\textbf{Proof:} By the independence assumption, for $m \geq 0$,\n\[
\text{Var}[\mu_{m+1}] = (1 - \alpha_m)^2 \text{Var}[\mu_m] + \alpha_m^2 \text{Var}[x_m] \\
\leq (1 - \alpha_m) \text{Var}[\mu_m] + \alpha_m^2.
\]
\begin{itemize}
\item We have $\alpha_m \to 0$ since $\sum_{m \geq 0} \alpha_m^2 < +\infty$.
\item Let $\epsilon > 0$ and suppose there exists $N \in \mathbb{N}$ such that for all $m \geq N$, $\text{Var}[\mu_m] \geq \epsilon$. Then, for $m \geq N$,
\[
\text{Var}[\mu_{m+1}] \leq \text{Var}[\mu_m] - \alpha_m \epsilon + \alpha_m^2,
\]
which implies $\text{Var}[\mu_{m+N}] \leq \text{Var}[\mu_N] - \epsilon \sum_{n=N}^{m+N} \alpha_n + \sum_{n=N}^{m+N} \alpha_n^2$,
\end{itemize}
contradicting $\text{Var}[\mu_{m+N}] \geq 0$.\n
Mean Estimation

- Thus, for all \( N \in \mathbb{N} \) there exists \( m_0 \geq N \) such that \( \text{Var}[\mu_{m_0}] < \epsilon \). Choose \( N \) large enough so that \( \forall m \geq N, \alpha_m \leq \epsilon \). Then,
  \[
  \text{Var}[\mu_{m_0+1}] \leq (1 - \alpha_{m_0})\epsilon + \epsilon \alpha_{m_0} = \epsilon.
  \]
- 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 $V$

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

$$= E_{s'}[r(s, \pi(s)) + \gamma V_\pi(s')|s].$$

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

$$V(s) \leftarrow (1 - \alpha)V(s) + \alpha[r(s, \pi(s)) + \gamma V(s')]$$

$$= V(s) + \alpha[r(s, \pi(s)) + \gamma V(s') - V(s)],$$

  temporal difference of $V$ values
**TD(0) Algorithm**

\[
\text{TD(0)}() \\
1 \quad V \leftarrow V_0 \triangleright \text{initialization.} \\
2 \quad \text{for } t \leftarrow 0 \text{ to } T \text{ do} \\
3 \quad \quad s \leftarrow \text{SELECTSTATE()} \\
4 \quad \quad \text{for each step of epoch } t \text{ do} \\
5 \quad \quad \quad r' \leftarrow \text{REWARD}(s, \pi(s)) \\
6 \quad \quad \quad s' \leftarrow \text{NEXTSTATE}(\pi, s) \\
7 \quad \quad \quad V(s) \leftarrow (1 - \alpha)V(s) + \alpha[r' + \gamma V(s')] \\
8 \quad \quad \quad s \leftarrow s' \\
9 \quad \text{return } V
\]
Q-Learning Algorithm

- **Idea:** assume deterministic rewards.

\[
Q^*(s, a) = \mathbb{E}[r(s, a)] + \gamma \sum_{s' \in S} \Pr[s' | s, a] V^*(s')
\]

\[
= \mathbb{E}[r(s, a) + \gamma \max_{a' \in A} Q^*(s', a')]
\]

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

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

(Watkins, 1989; Watkins and Dayan 1992)

Q-LEARNING($\pi$)

1. $Q \leftarrow Q_0 \; \triangleright \text{initialization, e.g., } Q_0 = 0.$
2. for $t \leftarrow 0$ to $T$ do
3.     $s \leftarrow \text{SELECTSTATE()}$
4.     for each step of epoch $t$ do
5.         $a \leftarrow \text{SELECTACTION}(\pi, s) \; \triangleright \text{policy } \pi \text{ derived from } Q, \text{ e.g., } \varepsilon\text{-greedy.}$
6.         $r' \leftarrow \text{REWARD}(s, a)$
7.         $s' \leftarrow \text{NEXTSTATE}(s, a)$
8.         $Q(s, a) \leftarrow Q(s, a) + \alpha [r' + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
9.         $s \leftarrow s'$
10. 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.

- 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.
Policies

- **Epsilon-greedy strategy:**
  - with probability $1 - \epsilon$ greedy action from $s$;
  - with probability $\epsilon$ random action.

- **Epoch-dependent strategy (Boltzmann exploration):**

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

  - $\tau_t \rightarrow 0$: greedy selection.
  - larger $\tau_t$: 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.
SARSA: On-Policy Algorithm

SARSA(\(\pi\))

1. \(Q \leftarrow Q_0 \triangleright \text{initialization, e.g., } Q_0 = 0.\)
2. \(\text{for } t \leftarrow 0 \text{ to } T \text{ do}\)
3. \(s \leftarrow \text{SelectState()}\)
4. \(a \leftarrow \text{SelectAction}(\pi(Q), s) \triangleright \text{policy } \pi \text{ derived from } Q, \text{ e.g., } \epsilon\text{-greedy.}\)
5. \(\text{for each step of epoch } t \text{ do}\)
6. \(r' \leftarrow \text{Reward}(s, a)\)
7. \(s' \leftarrow \text{NextState}(s, a)\)
8. \(a' \leftarrow \text{SelectAction}(\pi(Q), s') \triangleright \text{policy } \pi \text{ derived from } Q, \text{ e.g., } \epsilon\text{-greedy.}\)
9. \(Q(s, a) \leftarrow Q(s, a) + \alpha_t(s, a) [r' + \gamma Q(s', a') - Q(s, a)]\)
10. \(s \leftarrow s'\)
11. \(a \leftarrow a'\)
12. \(\text{return } Q\)
Notes

- Differences with Q-learning:
  - two states: current and next states.
  - maximum reward for next state not used for next state, instead new action.

- SARSA: name derived from sequence of updates.
TD(\(\lambda\)) Algorithm

- **Idea:**
  - TD(0) or Q-learning only use immediate reward.
  - use multiple steps ahead instead, for \(N\) steps:
    \[
    R^n_t = r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^{N-1} r_{t+N} + \gamma^N V(s_{t+N})
    \]
    \[
    V(s) \leftarrow V(s) + \alpha (R^n_t - V(s)).
    \]
  - TD(\(\lambda\)) uses \(R^\lambda_t = (1 - \lambda) \sum_{n=0}^{\infty} \lambda^n R^n_t\).

- **Algorithm:**
  \[
  V(s) \leftarrow V(s) + \alpha (R^\lambda_t - V(s)).
  \]
TD($\lambda$) Algorithm

TD($\lambda$)()
1 \textbf{V} \leftarrow V_0 \triangleright \text{initialization.}
2 \textbf{e} \leftarrow 0
3 \textbf{for } t \leftarrow 0 \textbf{ to } T \textbf{ do}
4 \hspace{1em} \textbf{s} \leftarrow \text{SELECTSTATE()}
5 \hspace{1em} \textbf{for } \text{each step of epoch } t \textbf{ do}
6 \hspace{2em} \textbf{s}' \leftarrow \text{NEXTSTATE}(\pi, s)
7 \hspace{1em} \delta \leftarrow r(s, \pi(s)) + \lambda V(s') - V(s)
8 \hspace{1em} e(s) \leftarrow \lambda e(s) + 1
9 \hspace{1em} \textbf{for } u \in S \textbf{ do}
10 \hspace{2em} \textbf{if } u \neq s \textbf{ then}
11 \hspace{3em} e(u) \leftarrow \gamma \lambda e(u)
12 \hspace{1em} V(u) \leftarrow V(u) + \alpha \delta e(u)
13 \hspace{1em} s \leftarrow s'
14 \textbf{return } V
TD-Gammon

(Tesauro, 1995)

- Large state space or costly actions: use regression algorithm to estimate $Q$ for unseen values.

- Backgammon:
  - large number of positions: 30 pieces, 24-26 locations,
  - large number of moves.

  - non-linear form of TD($\lambda$), 1.5M games played,
  - almost as good as world-class humans (master level).
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: \( \epsilon \)-greedy: play arm with best empirical reward with probability \( 1 - \epsilon_t \), random arm with probability \( \epsilon_t \).
Exponentially Weighted Average

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

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

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

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:

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


References


Appendix
Stochastic Approximation

- **Problem**: find solution of \( x = H(x) \) with \( x \in \mathbb{R}^N \) while
  - \( H(x) \) cannot be computed, e.g., \( H \) not accessible;
  - i.i.d. sample of noisy observations \( H(x_i) + w_i \), available, \( i \in [1, m] \), with \( E[w] = 0 \).

- **Idea**: algorithm based on iterative technique:
  \[
  x_{t+1} = (1 - \alpha_t)x_t + \alpha_t[H(x_t) + w_t]
  = x_t + \alpha_t[H(x_t) + w_t - x_t].
  \]
  
  - more generally \( x_{t+1} = x_t + \alpha_t D(x_t, w_t) \).
Supermartingale Convergence

**Theorem:** let $X_t, Y_t, Z_t$ be non-negative random variables such that $\sum_{t=0}^{\infty} Y_t < \infty$. If the following condition holds: $\mathbb{E} \left[ X_{t+1} \mid \mathcal{F}_t \right] \leq X_t + Y_t - Z_t$, then,

- $X_t$ converges to a limit (with probability one).
- $\sum_{t=0}^{\infty} Z_t < \infty$. 
Convergence Analysis

Convergence of $x_{t+1} = x_t + \alpha_t D(x_t, w_t)$, with history $F_t$ defined by

$$F_t = \{ (x_{t'})_{t' \leq t}, (\alpha_{t'})_{t' \leq t}, (w_{t'})_{t' < t} \}.$$ 

Theorem: let $\Psi : x \rightarrow \frac{1}{2} \| x - x^* \|^2_2$ for some $x^*$ and assume that

- $\exists K_1, K_2 : \mathbb{E} \left[ \left\| D(x_t, w_t) \right\|^2_2 \bigg| F_t \right] \leq K_1 + K_2 \Psi(x_t);$  
- $\exists c : \nabla \Psi(x_t) ^\top \mathbb{E} \left[ D(x_t, w_t) \bigg| F_t \right] \leq -c \Psi(x_t);$
  $$\alpha_t > 0, \sum_{t=0}^{\infty} \alpha_t = \infty, \sum_{t=0}^{\infty} \alpha_t^2 < \infty.$$  

Then, $x_t \xrightarrow{a.s.} x^*.$
Convergence Analysis

Proof: since $\Psi$ is a quadratic function,

$$\Psi(x_{t+1}) = \Psi(x_t) + \nabla \Psi(x_t)^\top (x_{t+1} - x_t) + \frac{1}{2} (x_{t+1} - x_t)^\top \nabla^2 \Psi(x_t) (x_{t+1} - x_t).$$

Thus,

$$E \left[ \Psi(x_{t+1}) | F_t \right] = \Psi(x_t) + \alpha_t \nabla \Psi(x_t)^\top E \left[ D(x_t, w_t) | F_t \right] + \frac{\alpha_t^2}{2} E \left[ \|D(x_t, w_t)\|^2 | F_t \right]$$

$$\leq \Psi(x_t) - \alpha_t c \Psi(x_t) + \frac{\alpha_t^2}{2} (K_1 + K_2 \Psi(x_t))$$

$$= \Psi(x_t) - \frac{\alpha_t^2}{2} K_1 - \left(\alpha_t c - \frac{\alpha_t^2 K_2}{2}\right) \Psi(x_t).$$

By the supermartingale convergence theorem, $\Psi(x_t)$ converges and $\sum_{t=0}^{\infty} \left(\alpha_t c - \frac{\alpha_t^2 K_2}{2}\right) \Psi(x_t) < \infty$.

Since $\alpha_t > 0$, $\sum_{t=0}^{\infty} \alpha_t = \infty$, $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$, $\Psi(x_t)$ must converge to 0.