Markov Decision Processes and Reinforcement Learning

- Markov Decision Process (compute optimal decisions offline)
- Reinforcement Learning (learn optimal decisions online)
  - Q-Learning
  - Upper Confidence Reinforcement Learning
Example 1: Chess Players

\( B \) often plays chess against \( C \) with the following odds.

When \( B \) plays aggressive (A): 55 % losses and 45 % wins.
When \( B \) plays defensive (D): 15 % losses, 75 % draws, 10 % wins.

\( B \) challenges \( C \) over a two game match.
Example 1: Chess Players

B often plays chess against C with the following odds.

When B plays aggressive (A): 55 % losses and 45 % wins.
When B plays defensive (D): 15 % losses, 75 % draws, 10 % wins.

B challenges C over a two game match.

Do you bet on B?
Example 1: Chess Players

B often plays chess against C with the following odds.

When B plays aggressive (A): 55 % losses and 45 % wins.
When B plays defensive (D): 15 % losses, 75 % draws, 10 % wins.

B challenges C over a two game match.

Do you bet on B?

All strategies AA, AD, DA, DD are losing. Respective expected gain −10%, −11%, −11% or −8% (for DD).
Example 1: Chess Players (II)

\( B \) uses the catenaccio strategy:

*play A in the first game then play D after a win or A after a defeat*

let us compute the winning probabilities:

Victory: WW ou WD: \( 0.45 \times (0.10 + 0.75) = 0.3825 \)

Draw: WL ou LW: \( 0.45 \times 0.15 + 0.55 \times 0.45 = 0.315 \)

Loss: LL: \( 0.55 \times 0.55 = 0.3025 \)

Expected gain: +8%.
Example 2: wheel of fortune

You have 4 tries at the wheel of fortune: (with gains 1,2,3,4,5,6,7,8,9,10)
Which strategy will you use to maximize your gain?
Example 2: wheel of fortune

You have 4 tries at the wheel of fortune: (with gains 1,2,3,4,5,6,7,8,9,10)
Which strategy will you use to maximize your gain?

Backward Induction
Last try: expected gain: 5.5

Last two tries: optimal strategy is to stop after first try if gain is > 5.5
Expected gain: \( \frac{1}{2} \times 5.5 + \frac{1}{2} \times 8 = 6.75 \)

Last 3 tries: optimal strategy is to stop after first try if gain is > 6.75
expected gain: \( 8.5 \times \frac{4}{10} + 6.75 \times \frac{6}{10} = 7.45 \)

4 tries: optimal strategy is to stop after first try if gain is > 7.45
expected gain: 7.915.
Markov Decision Process (MDP)

- States $x \in \mathcal{X}$ (finite)
- Horizon: $T$ (finite) or discounted or average.
- Rewards: $r_t(x)$
- Actions: $a \in \mathcal{A}$
- Transitions $P_t(x, a, y)$
- Strategy: $\pi: \mathcal{X} \rightarrow \mathcal{A}$

The goal of the controller is to maximize the expected gain over a finite horizon $T$

$$J(x) = \mathbb{E} \sum_{t=0}^{T} r_t(X_t, A_t)$$

with $X_0 = x$ et $X_{t+1}$ the next state according to the transition matrix $P_t(X_t, A_t, \cdot)$. 
Bellman Equation

Let us denote by $J_t^\pi(x)$, the expected gain under $\pi$ from $t$ to $T$, starting in $x$ at $t$.

$$J_t^\pi(x) = \mathbb{E} \sum_{u=t}^T r_u(X_u, \pi_u(X_u))$$

$$= r_t(x, \pi_t(x)) + \mathbb{E} \sum_{u=t+1}^T r_u(X_u, \pi_u(X_u))$$
Bellman Equation

Let us denote by $J_t^\pi(x)$, the expected gain under $\pi$ from $t$ to $T$, starting in $x$ at $t$.

$$J_t^\pi(x) = \mathbb{E} \sum_{u=t}^{T} r_u(X_u, \pi_u(X_u))$$

$$= r_t(x, \pi_t(x)) + \mathbb{E} \sum_{u=t+1}^{T} r_u(X_u, \pi_u(X_u))$$

$$= r_t(x, \pi_t(x)) + \mathbb{E} J_{t+1}^\pi(X_{t+1})$$
Bellman Equation

Let us denote by $J_t^\pi(x)$, the expected gain under $\pi$ from $t$ to $T$, starting in $x$ at $t$.

$$J_t^\pi(x) = \mathbb{E} \sum_{u=t}^{T} r_u(X_u, \pi_u(X_u))$$

$$= r_t(x, \pi_t(x)) + \mathbb{E} \sum_{u=t+1}^{T} r_u(X_u, \pi_u(X_u))$$

$$= r_t(x, \pi_t(x)) + \mathbb{E} J_{t+1}^\pi(X_{t+1})$$

$$= r_t(x, \pi_t(x)) + \sum_{y} P_t(x, \pi_t(x), y) J_{t+1}^\pi(y).$$
Bellman Optimality Equation

Let $J^*_t(x) = \sup_{\pi} J_\pi^t(x)$
Bellman Optimality Equation

Let $J_t^*(x) = \sup_{\pi} J_t^\pi(x)$

By backward induction on $t$,

$$J_t^*(x) = \sup_a \left( r_t(x, a) + \sum_y P_t(x, a, y) J_{t+1}^*(y) \right).$$

When $\sup$ is reached ($\mathcal{A}$ compact, $P$ et $r$ sont continuous en $a$), then the optimal strategy exists:

$$\pi_t^*(x) = \arg \max_a \left( r_t(x, a) + \sum_y P_t(x, a, y) J_{t+1}^*(y) \right).$$
Bellman Optimality Equation

Let \( J^*_t(x) = \sup_{\pi} J^\pi_t(x) \)

By backward induction on \( t \),

\[
J^*_t(x) = \sup_a \left( r_t(x, a) + \sum_y P_t(x, a, y) J^*_{t+1}(y) \right).
\]

When \( \sup \) is reached (\( \mathcal{A} \) compact, \( P \) et \( r \) sont continuous en \( a \)), then the optimal strategy exists:

\[
\pi^*_t(x) = \arg \max_a \left( r_t(x, a) + \sum_y P_t(x, a, y) J^*_{t+1}(y) \right).
\]

Computation using Dynamic Programming

1. \( J^*_T(x) = 0, \forall x \in \mathcal{X} \)
2. for \( t \) from \( T - 1 \) to 0 compute \( J^*_t \) using BOE.
Infinite Horizon

Discounted gain: \( J(x) = \mathbb{E} \sum_{t=0}^{\infty} \lambda^t r(X_t, A_t). \)

Transition matrices \( P(\cdot, a, \cdot) \) and rewards \( r(\cdot, a) \) are homogeneous in time.

Let us define \( J^T_t(x) = \mathbb{E} \sum_{u=t}^{T} \lambda^{u-t} r(X_u, A_u). \)

Bellman Equation in finite time is

\[
J^T_t(x) = \sup_a \left( r(x, a) + \lambda \sum_y P(x, a, y) J^T_{t+1}(y) \right).
\]

when \( T \) goes to \( \infty \)

\[
J^*(x) = \sup_a \left( r(x, a) + \lambda \sum_y P(x, a, y) J^*(y) \right).
\]
Dynamic Programming Operators

Operators $\mathcal{T}^\pi$ et $\mathcal{T}$:

$$L^\pi : \mathbb{R}^X \rightarrow \mathbb{R}^X$$
$$J(x) \rightarrow r(x, \pi(x)) + \lambda \sum_y P(x, \pi(x), y) J(y).$$

$$\mathcal{L} : \mathbb{R}^X \rightarrow \mathbb{R}^X$$
$$J(x) \rightarrow \sup_a \left( r(x, a) + \lambda \sum_y P(x, a, y) J(y) \right),$$
Value/Policy Iteration

Value Iteration

1. $J(x) = 0 \forall x \in X$
2. Repeat until $\varepsilon$-convergence $J_{n+1} := \mathcal{L}J_n$
3. $\pi := \arg \max \mathcal{T}J$

Policy Iteration

1. $\pi$ arbitrary policy.
2. Repeat until convergence of $\pi$:
3. $(J_{n+1} = L^{\pi_n}J_{n+1})$;
4. $\pi_{n+1} := \arg \max \mathcal{L}J_{n+1}$;
Average Reward

Average reward: \( g(x) = E \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{\infty} r(X_t, A_t) \).

Under some mild conditions (weakly communicating MDP) the limit exists and the optimal average reward does not depend on \( x \).

\( g^* \) is the solution of the non discounted Bellman equation:

\[
g^* + h(i) = \max_a \left( r(x, a) + \sum_j P(i, a, j) h(j) \right)
\]

Can be computed using value iteration, policy iteration or linear programming.
MDP suffers from two main weaknesses:

- **Curse of dimensionality**: the state space of Markovian models are large even for moderate size systems.

- **Curse of Information**: The transition probabilities $P(x, a, y)$ must be known for all $x, a, y$. Actually there are close relations between these two problems: Deep Reinforcement Learning is an efficient way to address both drawbacks at the same time.
MDP suffers from two main weaknesses:

- Curse of dimensionality: the state space of Markovian models are large even for moderate size systems.
- Curse of Information: The transition probabilities $P(x, a, y)$ must be known for all $x, a, y$.

Actually there are close relations between these two problems: Deep Reinforcement Learning is an efficient way to address both drawbacks at the same time.
Stochastic Approximation

(Also called the ODE Method)
A stochastic dynamical system of the form

\[ X_{n+1} = X_n + \gamma_n(f(X_n) + M_{n+1}) \]

is a stochastic approximation of the o.d.e. \( \dot{x} = f(x) \) if \( E(M_{n+1}|F_n) = 0, E(||M_{n+1}||^2|F_n) \leq K(X_n + C) \) and \( \gamma_n \) is in \( L^2 \setminus L^1 \).

If \( \sup ||X_n|| \) is bounded, then \( X_n \) is an APT (Asymptotic Pseudo Trajectory) of the o.d.e., (Benaim):

\[
\lim_{s \to \infty} \sup_{t \in [s, s+T]} ||X(t) - x_s(t)|| = 0.
\]

Furthermore, \( X_n \) converges to an internally chain transitive invariant set of the o.d.e..

Corollary

If the o.d.e. has a unique global attractor, then \( \lim_{n \to \infty} f(X_n) = 0 \) a.s.
Consider a discounted MDP \((r, A, P, \beta)\). By definition, the Q-value function \(q^*(x, a)\) is the optimal gain starting in state \(x\) and using action \(a\). The optimal reward is related to \(q^*\) by \(J^*(x) = \max_a q^*(x, a)\).

The BOE for \(Q\) is:

\[
q^*(x, a) = r(x, a) + \beta \sum_y P(x, a, y)J^*(y)
\]

\[
= \sum_y P(x, a, y)\left[ r(x, a) + \beta \max_b q^*(y, b) \right]
\]

In vector form, \(q^* = Fq^*\).

Value Iteration for \(Q\)-value function: \(q_{n+1} = Fq_n\) converges to the optimal \(q^*\) because \(F\) is \(\beta\)-contracting.
Q-learning Algorithm (PhD of Watkins, 1989)

Learning the optimal actions in each state by monitoring the system on-line:
In state $X_n$, choose an action $a_n$ according to some selection policy. The system gives you an instant reward $r$ and moves to state $X_{n+1}$ according to an (unknown) Markov transition.

$Q_{n+1}(x, a) = (1 - \gamma_n) Q_n(x, a) + \gamma_n (r(x, a) + \beta \max_b Q_n(x_{n+1}, b))$

$Q_{n+1}(x, a) = Q_n(x, a) \text{ if } (x, a) \neq (X_n, a_n)$
Q-learning Algorithm (PhD of Watkins, 1989)

Learning the optimal actions in each state by monitoring the system on-line:
In state $X_n$, choose an action $a_n$ according to some selection policy. The system gives you an instant reward $r$ and moves to state $X_{n+1}$ according to an (unknown) Markov transition.

Given this sequence $(X_n, a_n)_{n \in \mathbb{N}}$ of state-action pairs obtained by monitoring the system, the Q-learning algorithm computes the following on-line sequence of values for all states and actions:

$$Q_{n+1}(X_n, a_n) = (1 - \gamma_n)Q_n(X_n, a_n) + \gamma_n(r(X_n, a_n) + \beta \max_b Q_n(X_{n+1}, b))$$

$$Q_{n+1}(x, a) = Q_n(x, a) \quad \text{if} \quad (x, a) \neq (X_n, a_n)$$
Convergence of Q-learning Algorithm

Theorem

If each state-action couple \((x, a)\) appears an infinite number of times in the sequence \((X_n, a_n)\), then \(\lim_{n \to \infty} Q_n(x, a) = q^*(x, a)\).

The stochastic sequence

\[
Q_{n+1}(x, a) = Q_n(x, a) + \gamma_n \left[ r(x, a) + \beta \max_b Q_n(Y, b) - Q_n(x, a) \right]
\]

is a stochastic approximation of

\[
\dot{q}(x, a) = \mathbb{E} \left[ r(x, a) + \beta \max_b q(Y, b) - q(x, a) \bigg| (x, a) \right]
\]

The operator \(F\) is contracting, therefore, \(Q_{n+1}\) converges to the unique point \(q^\infty\) s.t.

\[
\mathbb{E} \left[ r(x, a) + \beta \max_b q^\infty(Y, b) - q^\infty(x, a) \bigg| (x, a) \right] = 0.
\]

so \(q^\infty = q^*\).
Speed of Convergence: Theoretical Bounds

Several variants of the algorithm exist according to the choice of $a_n$. The classical $\varepsilon$ greedy selection: current best action w. p. $1 - \varepsilon$ (other choices such as exp-weight).

The convergence rate of Q-learning has been investigated by Even-Dar and Mansour, 2003.

**Theorem**

For any $\varepsilon > 0$, after

$$T = O \left( \frac{R^2 \log \frac{N}{\delta}}{\varepsilon^2 (1 - \beta)^4} \right)$$

steps of QL, the uniform approximation error $||q^* - Q_T|| \leq \varepsilon$, with probability larger than $1 - \delta$.

Essential points: convergence in $\sqrt{T}$, in $\log(N)$, in $(1 - \beta)^4$. 
In most experiments we tried, convergence is slow. Example from Kimang’s work on restful bandits (4 states, 2 actions, $\beta = 0.5$)

Many improvements have been proposed: ZAP Q-learning, speedy Q-learning, Approximate/Projected Q-learning (Berstekas)…
Upper Confidence Reinforcement Learning

Principle: Use UCB ideas to learn the optimal policy.
Let us consider a non-discounted unichain MDP \((M)\),

**Repeat**

1. estimate: \(\hat{r}_t(x, a) = \frac{R_t(x, a)}{N_t(x, a)}\) and \(\hat{p}_t(x, a, y) = \frac{N_t(x, a, y)}{N_t(x, a)}\)

2. Compute new policy: \(\tilde{\pi}_k := \arg \max_{\pi} (g(M, \pi), M' \in \mathcal{M}_t)\) (L.P.)

   over all \(|r'(x, a) - \hat{r}_t(x, a)| \leq UCB_r\) and

   \(|p'(x, a, y) - \hat{p}_t(x, a, y)| \leq UCB_p\)

3. Use policy \(\tilde{\pi}_k\) until \(UCB_r\) or \(UCB_p\) decreases by half.

with \(UCB_r(t, x, a) = \sqrt{\frac{\log(2t^2NA)}{2N_t(x, a)}}\). (similar definition for \(UCB_p\)).
A direct consequence of Chernoff-Hoeffding inequality is that $M$ belongs to $\mathcal{M}_t$ with probability higher than $1 - 1/t^2$.

Expected regret: 
\[
R_T = \mathbb{E} \sum_{t=1}^{T} r_t - Tg^*
\]

\[
R_T \leq C \frac{AN^5 T_M K_M^2}{\Delta^2} \log T
\]

where $T_M$ is the max first passage time of the MDP $M$

$K_M$ is the ratio of first passage time to return time.

and $\Delta$ is the min cost difference over all suboptimal policies.
That’s all Folks!