Introduction to Reinforcement Learning and multi-armed bandits

Rémi Munos

INRIA Lille - Nord Europe
Currently on leave at MSR-NE
http://researchers.lille.inria.fr/~munos/

NETADIS Summer School 2013, Hillerod, Denmark
Outline of the course

- Part 1: Introduction to Reinforcement Learning and Dynamic Programming
  - Dynamic programming: value iteration, policy iteration
  - Q-learning.

- Part 2: Approximate DP and RL
  - $L_\infty$-norm performance bounds
  - Sample-based algorithms.
  - Links with statistical learning

- Part 3: Intro to multi-armed bandits
  - The stochastic bandit: UCB
  - The adversarial bandit: EXP3
  - Approximation of Nash equilibrium
  - Monte-Carlo Tree Search
Part 1: Introduction to Reinforcement Learning and Dynamic Programming

A few general references:

Introduction to Reinforcement Learning (RL)

- Learn to make good decisions in unknown environments
- Learning from experience: success or failures
- Examples: learning to ride a bicycle, play chess, autonomous robotics, operation research, playing in stochastic market, ...

I learned to ride with RL...
A few applications

- KnightCap [Baxter et al. 1998]: chess ($\approx 2500$ ELO)
- Robotics: juggling, acrobots [Schaal and Atkeson, 1994]
- Mobile robot navigation [Thrun et al., 1999]
- Elevator controller [Crites et Barto, 1996],
- Packet Routing [Boyan et Littman, 1993],
- Job-Shop Scheduling [Zhang et Dietterich, 1995],
- Production manufacturing optimization [Mahadevan et al., 1998],
- Game of poker (Bandit algo for Nash computation)
- Game of go (hierarchical bandits, UCT)

http://www.ualberta.ca/~szepesva/RESEARCH/RLApplications.html
**Reinforcement Learning**

- **Environment**: can be stochastic (Tetris), adversarial (Chess), partially unknown (bicycle), partially observable (robot)
- **Available information**: the reinforcement (may be delayed)
- **Goal**: maximize the expected sum of future rewards.

**Problem**: How to sacrifice a short term small reward to privilege larger rewards in the long term?
Optimal value function

- Gives an evaluation of each state if the agent plays optimally.
- Ex: in a stochastic environment:

\[ V^*(x_t) \]

Transition probabilities

0.3 0.5 0.2

Bellman equation:

\[
V^*(x_t) = \max_{a \in A} \left[ r(x_t, a) + \sum_y p(y|x_t, a) V^*(y) \right]
\]

Temporal difference:

\[
\delta_t = V^*(x_{t+1}) + r(x_t, a_t) - V^*(x_t)
\]

If \( V^* \) is known, then when choosing the optimal action \( a_t \), \( \mathbb{E}[\delta_t] = 0 \) (i.e., in average there is no surprise)
Challenges of RL

- Environment may be stochastic, adversarial, partially observable...
- The state-dynamics and reward functions are unknown: we need to combine
  - Learning
  - Planning
- The curse of dimensionality: We need to rely on *approximations* for representing the value function and the optimal policy.
Introduction to Dynamic Programming

A Markov Decision Process \((X, A, p, r)\) defines a discrete-time process \((x_t) \in X\) where:

- \(X\): state space
- \(A\): action space (or decisions)
- State dynamics: All relevant information about future is included in the current state and action (Markov property)

\[
P(x_{t+1} \mid x_t, x_{t-1}, \ldots, x_0, a_t, a_{t-1}, \ldots, a_0) = P(x_{t+1} \mid x_t, a_t)
\]

Thus we define the transition probabilities \(p(y|x, a)\)

- Reinforcement (or reward): \(r(x, a)\) is obtained when choosing action \(a\) in state \(x\).
Definition of policy

Policy $\pi = (\pi_1, \pi_2, \ldots)$, where at time $t$,

$$\pi_t : X \rightarrow A$$

maps an action $\pi_t(x)$ to any possible state $x$.

Given a policy $\pi$ the process $(x_t)_{t \geq 0}$ is a Markov chain with transition probabilities

$$p(x_{t+1}|x_t) = p(x_{t+1}|x_t, \pi_t(x_t)).$$

When the policy is independent of time, $\pi = (\pi, \pi, \ldots, \pi)$, the policy is called stationary (or Markovian).
Performance of a policy

For any policy $\pi$, define the value function $V^\pi$:

**Infinite horizon:**

- **Discounted:** $V^\pi(x) = \mathbb{E}\left[ \sum_{t=0}^{\infty} \gamma^t r(x_t, a_t) \mid x_0 = x; \pi \right]$, where $0 < \gamma < 1$ is the discount factor
- **Undiscounted:** $V^\pi(x) = \mathbb{E}\left[ \sum_{t=0}^{\infty} r(x_t, a_t) \mid x_0 = x; \pi \right]$
- **Average:** $V^\pi(x) = \lim_{T \to \infty} \frac{1}{T} \mathbb{E}\left[ \sum_{t=0}^{T-1} r(x_t, a_t) \mid x_0 = x; \pi \right]$

**Finite horizon:** $V^\pi(x, t) = \mathbb{E}\left[ \sum_{s=t}^{T-1} r(x_s, a_s) + R(x_T) \mid x_t = x; \pi \right]$
The dilemma of the Netadis SS student

You try to maximize the sum of rewards!
Solution of the Netadis SS student

\[ V_5 = -10, \ V_6 = 100, \ V_7 = -1000, \]
\[ V_4 = -10 + 0.9 V_6 + 0.1 V_4 \approx 88.9. \]
\[ V_3 = -1 + 0.5 V_4 + 0.5 V_3 \approx 86.9. \]
\[ V_2 = 1 + 0.7 V_3 + 0.3 V_1 \text{ and } \]
\[ V_1 = \max \{0.5 V_2 + 0.5 V_1, 0.5 V_3 + 0.5 V_1\}, \text{ thus: } V_1 = V_2 = 88.3. \]
Infinite horizon, discounted problems

For any stationary policy \( \pi \), define the value function \( V^\pi \) as:

\[
V^\pi(x) = \mathbb{E}\left[ \sum_{t=0}^{\infty} \gamma^t r(x_t, \pi(x_t)) \mid x_0 = x; \pi \right],
\]

where \( 0 \leq \gamma < 1 \) a discount factor (which relates rewards in the future compared to current rewards).
Bellman equation for $V^\pi$

**Proposition 1 (Bellman equation).**

For any policy $\pi$, $V^\pi$ satisfies:

$$V^\pi(x) = r(x, \pi(x)) + \gamma \sum_{y \in X} p(y|x, \pi(x)) V^\pi(y),$$

Thus $V^\pi$ is the fixed point of the **Bellman operator** $T^\pi$ (i.e., $V^\pi = T^\pi V^\pi$) where $T^\pi W$ is defined as

$$T^\pi W(x) = r(x, \pi(x)) + \gamma \sum_{y} p(y|x, \pi(x)) W(y)$$

Using matrix notations, $T^\pi W = r^\pi + \gamma P^\pi W$, where $r^\pi(x) = r(x, \pi(x))$ and $P^\pi(x, y) = p(y|x, \pi(x))$. 
Proof of Proposition 1

\[ V^\pi(x) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r(x_t, \pi(x_t)) \mid x_0 = x; \pi \right] \]

\[ = r(x, \pi(x)) + \mathbb{E} \left[ \sum_{t \geq 1} \gamma^t r(x_t, \pi(x_t)) \mid x_0 = x; \pi \right] \]

\[ = r(x, \pi(x)) + \gamma \sum_{y} P(x_1 = y \mid x_0 = x; \pi) \]

\[ \mathbb{E} \left[ \sum_{t \geq 1} \gamma^{t-1} r(x_t, \pi(x_t)) \mid x_1 = y; \pi \right] \]

\[ = r(x, \pi(x)) + \gamma \sum_{y} p(y \mid x, \pi(x)) V^\pi(y). \]
Bellman equation for $V^*$

Define the **optimal value function**: $V^* = \sup_\pi V^\pi$.

**Proposition 2 (Dynamic programming equation).**

$V^*$ satisfies:

$$V^*(x) = \max_{a \in A} \left[ r(x, a) + \gamma \sum_{y \in X} p(y|x, a) V^*(y) \right].$$

Thus $V^*$ is the fixed point of the **Dynamic programming operator** $\mathcal{T}$ (i.e., $V^* = \mathcal{T} V^*$) where $\mathcal{T} W$ is defined as

$$\mathcal{T} W(x) = \max_{a \in A} \left[ r(x, a) + \gamma \sum_{y \in X} p(y|x, a) W(y) \right].$$
Proof of Proposition 2

And for all policy \( \pi = (a, \pi') \) (not necessarily stationary),

\[
V^*(x) = \max_{\pi} \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r(x_t, \pi(x_t)) \mid x_0 = x; \pi \right]
\]

\[
= \max_{(a, \pi')} \left[ r(x, a) + \gamma \sum_y p(y|x, a) V^{\pi'}(y) \right]
\]

\[
= \max_a \left[ r(x, a) + \gamma \sum_y p(y|x, a) \max_{\pi'} V^\pi(y) \right] \quad (1)
\]

\[
= \max_a \left[ r(x, a) + \gamma \sum_y p(y|x, a) V^*(y) \right].
\]

where (1) holds since:

- \( \max_{\pi'} \sum_y p(y|x, a) V^{\pi'}(y) \leq \sum_y p(y|x, a) \max_{\pi'} V^{\pi'}(y) \)

- Let \( \bar{\pi} \) be the policy defined by \( \bar{\pi}(y) = \arg \max_{\pi'} V^{\pi'}(y) \). Thus \( \sum_y p(y|x, a) \max_{\pi'} V^{\pi'}(y) = \sum_y p(y|x, a) V^{\bar{\pi}}(y) \leq \max_{\pi'} \sum_y p(y|x, a) V^{\pi'}(y) \).
Properties of the Bellman operators

- **Monotonicity:** If $W_1 \leq W_2$ (componentwise) then
  \[
  T^\pi W_1 \leq T^\pi W_2, \text{ and } TW_1 \leq TW_2.
  \]

- **Contraction in max-norm:** For any vectors $W_1$ and $W_2$,
  \[
  \|T^\pi W_1 - T^\pi W_2\|_\infty \leq \gamma \|W_1 - W_2\|_\infty,
  \]
  \[
  \|TW_1 - TW_2\|_\infty \leq \gamma \|W_1 - W_2\|_\infty.
  \]

Indeed, for all $x \in X$,

\[
|TW_1(x) - TW_2(x)| = \left| \max_a \left[ r(x, a) + \gamma \sum_y p(y|x, a) W_1(y) \right]
\right.
\[
- \left| \max_a \left[ r(x, a) + \gamma \sum_y p(y|x, a) W_2(y) \right] \right|
\]
\[
\leq \gamma \max_a \sum_y p(y|x, a) |W_1(y) - W_2(y)|
\]
\[
\leq \gamma \|W_1 - W_2\|_\infty
\]
Properties of the value functions

**Proposition 3.**

1. \( V^\pi \) is the unique fixed-point of \( T^\pi \)

\[
V^\pi = T^\pi V^\pi.
\]

2. \( V^* \) is the unique fixed-point of \( T \):

\[
V^* = T V^*.
\]

3. For any policy \( \pi \), we have \( V^\pi = (I - \gamma P^\pi)^{-1} r^\pi \)

4. The policy defined by

\[
\pi^*(x) \in \arg \max_{a \in A} \left[ r(x, a) + \gamma \sum_{y} p(y|x, a) V^*(y) \right]
\]

is optimal (and stationary)
1. From Proposition 1, $V^\pi$ is a fixed point of $T^\pi$. Uniqueness comes from the contraction property of $T^\pi$.

2. Idem for $V^\ast$.

3. $V^\pi = T^\pi V^\pi = r^\pi + \gamma P^\pi V^\pi$. Thus $(I - \gamma P^\pi)V^\pi = r^\pi$. Now $P^\pi$ is a stochastic matrix (whose eigenvalues have a modulus $\leq 1$), thus the eigenvalues of $(I - \gamma P^\pi)$ have a modulus $\geq 1 - \gamma > 0$, thus is invertible.

4. From the definition of $\pi^\ast$, we have

$$T^{\pi^\ast} V^\ast = T V^\ast = V^\ast$$

Thus $V^\ast$ is the fixed-point of $T^{\pi^\ast}$. But, by definition, $V^{\pi^\ast}$ is the fixed-point of $T^{\pi^\ast}$ and since there is uniqueness of the fixed-point, $V^{\pi^\ast} = V^\ast$ and $\pi^\ast$ is optimal.
Value Iteration

Proposition 4.

- For any bounded $\pi$ and $V_0$, define $V_{k+1} = T^\pi V_k$. Then $V_k \to V^\pi$.
- For any bounded $V_0$, define $V_{k+1} = TV_k$. Then $V_k \to V^*$.

Proof.

$||V_{k+1} - V^*|| = ||TV_k - TV^*|| \leq \gamma ||V_k - V^*|| \leq \gamma^{k+1} ||V_0 - V^*|| \to 0$

(idem for $V^\pi$)

Variant: asynchronous iterations
Policy Iteration

Choose any initial policy $\pi_0$. Iterate:

1. **Policy evaluation**: compute $V^{\pi_k}$.

2. **Policy improvement**: $\pi_{k+1}$ greedy w.r.t. $V^{\pi_k}$:

$$
\pi_{k+1}(x) \in \arg \max_{a \in A} \left[ r(x, a) + \gamma \sum_{y} p(y|x, a) V^{\pi_k}(y) \right],
$$

(i.e. $\pi_{k+1} \in \arg \max_{\pi} T^\pi V^{\pi_k}$)

Stop when $V^{\pi_k} = V^{\pi_{k+1}}$.

**Proposition 5.**

Policy iteration generates a sequence of policies with increasing performance ($V^{\pi_{k+1}} \geq V^{\pi_k}$) and (in the case of finite state and action spaces) terminates in a finite number of steps with the optimal policy $\pi^*$. 

Proof of Proposition 5

From the definition of the operators \( \mathcal{T}, \mathcal{T}^{\pi_k}, \mathcal{T}^{\pi_{k+1}} \) and from \( \pi_{k+1} \),

\[
V^{\pi_k} = \mathcal{T}^{\pi_k} V^{\pi_k} \leq \mathcal{T} V^{\pi_k} = \mathcal{T}^{\pi_{k+1}} V^{\pi_k},
\]

and from the monotonicity of \( \mathcal{T}^{\pi_{k+1}} \), we have

\[
V^{\pi_k} \leq \lim_{n \to \infty} (\mathcal{T}^{\pi_{k+1}})^n V^{\pi_k} = V^{\pi_{k+1}}.
\]

Thus \( (V^{\pi_k})_k \) is a non-decreasing sequence. Since there is a finite number of possible policies (finite state and action spaces), the stopping criterion holds for a finite \( k \); We thus have equality in (2), thus

\[
V^{\pi_k} = \mathcal{T} V^{\pi_k}
\]

so \( V^{\pi_k} = V^* \) and \( \pi_k \) is an optimal policy.
Back to Reinforcement Learning

What if the transition probabilities $p(y|x, a)$ and the reward functions $r(x, a)$ are unknown?

In DP, we used their knowledge

- in value iteration:
  \[
  V_{k+1}(x) = \mathcal{T} V_k(x) = \max_a \left[ r(x, a) + \gamma \sum_y p(y|x, a) V_k(y) \right].
  \]

- in policy iteration:
  
  - when computing $V^{\pi_k}$ (which requires iterating $\mathcal{T}^{\pi_k}$)
  
  - when computing the greedy policy:
  \[
  \pi_{k+1}(x) \in \arg\max_{a \in A} \left[ r(x, a) + \gamma \sum_y p(y|x, a) V^{\pi_k}(y) \right],
  \]

RL = introduction of 2 ideas: **Q-functions** and **sampling**.
Definition of the Q-value function

Define the Q-value function $Q^\pi : X \times A \rightarrow \mathbb{R}$: for a policy $\pi$,

$$Q^\pi(x, a) = \mathbb{E}\left[ \sum_{t \geq 0} \gamma^t r(x_t, a_t) | x_0 = x, a_0 = a, a_t = \pi(x_t), t \geq 1 \right]$$

and the optimal Q-value function $Q^*(x, a) = \max_\pi Q^\pi(x, a)$.

**Proposition 6.**

$Q^\pi$ and $Q^*$ satisfy the Bellman equations:

$$Q^\pi(x, a) = r(x, a) + \gamma \sum_{y \in X} p(y|x, a) Q^\pi(y, \pi(y))$$

$$Q^*(x, a) = r(x, a) + \gamma \sum_{y \in X} p(y|x, a) \max_{b \in A} Q^\pi(y, b)$$

Idea: compute $Q^*$ and then $\pi^*(x) \in \arg \max_a Q^*(x, a)$. 
Q-learning algorithm [Watkins, 1989]

Builds a sequence of Q-value functions $Q_k$. Whenever a transition $x_t, a_t \xrightarrow{r_t} x_{t+1}$ occurs, update the Q-value:

$$Q_{k+1}(x_t, a_t) = Q_k(x_t, a_t) + \eta_k(x_t, a_t) \left[ r_t + \gamma \max_{b \in A} Q_k(x_{t+1}, b) - Q_k(x_t, a_t) \right].$$

**Proposition 7 (Watkins et Dayan, 1992).**

Assume that all state-action pairs $(x, a)$ are visited infinitely often and that the learning steps satisfy for all $x, a$,

$$\sum_{k \geq 0} \eta_k(x, a) = \infty, \quad \sum_{k \geq 0} \eta_k^2(x, a) < \infty,$$

then $Q_k \xrightarrow{a.s.} Q^*$. 

The proof relies on Stochastic Approximation for estimating the fixed-point of a contraction mapping.
Q-learning algorithm

Deterministic case, discount factor $\gamma = 0.9$. Take steps $\eta = 1$.

After transition $x, a \xrightarrow{r} y$ update $Q_{k+1}(x, a) = r + \gamma \max_{b \in A} Q_k(y, b)$
Optimal Q-values

Bellman's equation: $Q^*(x, a) = \gamma \max_{b \in A} Q^*(\text{next-state}(x, a), b)$. 
First conclusions

When the state-space is finite and “small”:

- If transition probabilities and rewards are known, then DP algorithms (value iteration, policy iteration) compute the optimal solution
- Otherwise, use sampling techniques and RL algorithms (Q-learning, TD(\(\lambda\))) apply

2 main issues:

- Usually state-space is large (infinite)! We need to build approximate solutions.
- We need to design clever exploration strategies.