# MDP and RL: $Q$-learning, stochastic approximation

Nicolas Gast

October 9, 2023

This document is a **DRAFT** of notes of the second part of the course on *MDP and reinforcement learning* given at ENS de Lyon during the academic years 2023-2024 and the forthcoming 2024-2025.

## Contents



Main references: [2] for Q-learning and variants (Section 1), and [1] (Section 2) for the stochastic approximation part. The rest is from research papers.

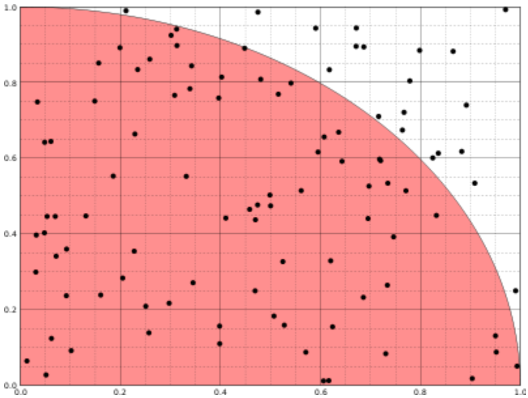
# 1 Monte-Carlo methods and Q-learning

Recall Bellman's equation:

$$V^*(s) = \max_{a \in \mathcal{A}} Q^*(s,a)$$
$$Q^*(s,a) = \mathbf{r}(s,\pi(s)) + \gamma \sum_{s'} V^*(s')p(s' \mid s,a)$$

Our assumption: we have access to a simulator.

## 1.1 Estimation via Monte-Carlo




Source: https://fr.wikipedia.org/wiki/Mthode_de_Monte-Carlo#Dtermination_de_la_valeur_de_%CF%80

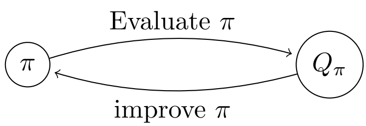
Figure 1: Estimation of $\pi$ via Monte-Carlo.

See Figure 1. Area is $\pi/4$. A point $(x,y)$ is in the red zone if $x^2 + y^2 \leq 1$. Estimation via rollout:

$$V^\pi(S_t) = \mathbb{E}\left[G_t \mid S_t = s, \pi\right].$$

- Monte-Carlo = sample $G_t$ by using rollout. Can use every-visit or first-visit.
- Converges in $O(1/\sqrt{n})$

### 1.1.1 Monte-Carlo optimzation



Recall: improve can be done by using *greedy*:

$$\pi(s) = \arg\max_{a \in \mathcal{A}} Q(s,a).$$

Possible problems:

- One may need many samples for all actions.
- Some action-pair might not be visited.

Solutions: exploration/exploitation tradeoff (previous), importance sampling.

## 1.2 TD-learning

Bellman's equation states:

$$\begin{aligned} V(S_t) &= \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \dots\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma V(S_{t+1})\right]. \end{aligned}$$

This is equivalent to

$$0 = \mathbb{E}\left[\underbrace{R_{t+1} + \gamma V(S_{t+1}) - V(S_t)}_{\text{TD error}}\right]$$

The TD learning algorithm uses the updates:

$$V(S_t) := V(S_t) + \alpha_t(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))),$$

where $\alpha$ is a learning rate such that $\sum_t \alpha_t = +\infty$ and $\sum_t (\alpha_t)^2 < \infty$.

*Proof.* Main proof: see later. for some ideas:
Let $\beta_t(s)$ be such that

$$\beta_t(s) = \begin{cases} 0 & \text{if } s = S_t \\ \alpha_t & \text{otherwise} \end{cases}$$

Let $V_t$ be the $V$-table at time $t$. The definition of $\beta_t$ implies that for all $s$:

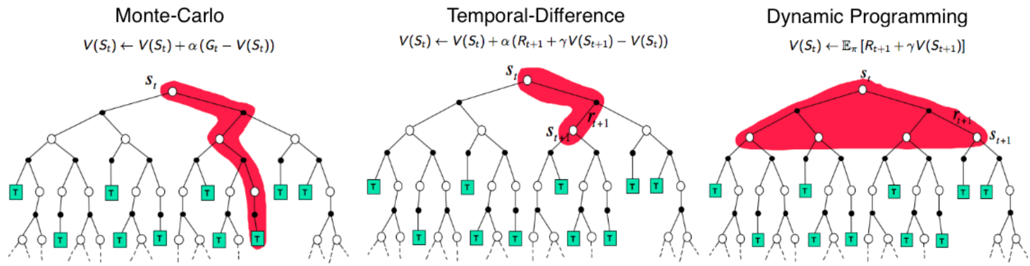
$$V_{t+1}(s) := V_t(s) + \beta_t(s)\left(\underbrace{R_{t+1} + \gamma V_t(S_{t+1})}_{=T^\pi V_t + \text{noise}} - V_t(s)\right).$$

with $\sum_t \beta_t(s) = \infty$ and $\sum_t \beta_t^2(s) < \infty$.

As $T^\pi$ is contracting, Theorem 1 of (*On the convergence of stochastic iterative dynamic programming algorithms., Jaakkola, Jordan, Singh, NeurIPS 93*) shows that this implies $\lim_{t\to\infty} V_t = V^\pi$ almost surely. □

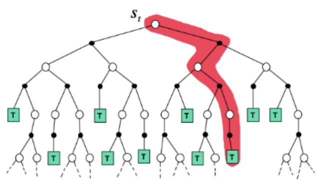
## 1.3 Relation between MC, TD and DP

$$
\begin{aligned}
V(S_t) &= \mathbb{E}\left[G_t\right] && MC \\
V(S_t) &= \mathbb{E}\left[R_{t+1} + \gamma V(S_{t+1})\right] && TD \\
V(S_t) &= \mathbb{E}\left[R_{t+1}\right] + \gamma \sum_{s'} V(S_{t+1})\mathbb{P}(S_{t+1} = s') && DP
\end{aligned}
$$

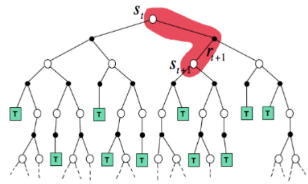



(figure from Sutton and Barto)

- MC simulates a full trajectory
- TD samples one-step and uses a previous estimation of $V$.
- DP needs all possible values of $V(s')$.



MC: One full trajectory for update



TD: Updates take time to propagate

The tradeoff comes by using TD($\lambda$):

- Use $n$-step returns (see Sutton-Barto, chapter 7).

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^{t+n} V(S_{t+n}).$$

- $TD(\lambda)$ (see Sutton-Barto, chapter 12 or Szepesvári, Section 2.1.3).

$$G_t(\lambda) = (1-\lambda) \sum_{n=1}^{T} \lambda^{n-1} G_{t:t+n} + \lambda^T G_t.$$

## 1.4 $Q$-learning and SARSA

Bellman's equations are:

$$
\begin{aligned}
V^\pi(S_t) &= \mathbb{E}^\pi\left[R_{t+1} + \gamma V^\pi(S_{t+1})\right] && \text{to evaluate } \pi \\
Q^*(S_t, A_t) &= \mathbb{E}\left[R_{t+1} + \gamma \max_a Q^*(S_{t+1}, a)\right] && \text{to find the best policy}
\end{aligned}
$$

This leads to two variant of:

- Q-learning = off-policy learning.
  - Choose $A_t \sim \pi$.
  - Apply TD-learning replacing $V(s)$ by $\max_a Q(s, a)$.
- SARSA = on-policy learning:
  - Choose $A_{t+1} \sim \arg\max_{a\in\mathcal{A}} Q(S_{t+1}, a)$.
  - Apply TD-learning replacing $V(s)$ by $Q(s, A_{t+1})$.

#### 1.4.1 $Q$-learning

$$
\begin{aligned}
&A_t \sim \pi \\
&Q(S_t, A_t) := Q(S_t, A_t) + \alpha_t \left( R_{t+1} + \gamma \max_{a\in\mathcal{A}} Q(S_{t+1}, a) - Q(S_t, A_t) \right).
\end{aligned}
$$

**Theorem 1.** *Assume that $\gamma < 1$ and that:*

- *Any station-action pair $(a, s)$ is visited infinitely often.*
- $\sum_t \alpha_t = \infty$ *and* $\sum_t \alpha_t^2 < \infty$.

*Then: $Q$ converges almost surely to the optimal $Q^*$-table as $t$ goes to infinity.*

#### 1.4.2 SARSA

SARSA (name comes from $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}$)

$$
\begin{aligned}
&A_{t+1} \sim \arg\max Q(S_t, A_t) \text{ (or } \varepsilon\text{-greedy)} \\
&Q(S_t, A_t) := Q(S_t, A_t) + \alpha_t \left( R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \right).
\end{aligned}
$$

**Open questions**:

- Does it converge (and why?)
- How to choose the step-size?
- How to explore?

# 2 Stochastic approximation

## 2.1 Introduction and example: the ODE method

$$x_{n+1} = x_n + a_n(f(x_n) + \text{noise}),$$

- TD-learning or Q-learning.

- Stochastic gradient descent. We are given $N$ couples $(X_1, Y_1) \dots (X_N, Y_N)$ and a parametric function $g_x$. We want to find $x$ such that $g_x(X_i) \approx Y_i$ for all $i$. We model this as an empirical risk minimization by using a loss function $\ell$:

$$F(x) = \frac{1}{N} \sum_{k=1}^{N} \ell(f_x(X_k), Y_k) = \mathbb{E}\left[\ell(f_x(X), Y)\right],$$

  where the expectation is taken uniformly over all data.

  We want to do $x_{n+1} = x_n - a_n \nabla_x F(x)$ but this is costly. The stochastic gradient descent is:

  - Pick $(X_n, Y_n)$ uniformly at random among all data points.
  - Computes $x_{n+1} - = a_n \nabla_x \ell(g_{x_n}(X), Y)$.

  This rewrites as:

$$x_{n+1} = x_n + a_n(f(x_n) + \text{noise}),$$

  where $f(x) = \nabla_x F(x)$.

In what follows, we want to show that the stochastic system behaves as the solutions of the $\dot{x} = f(x)$. This helps us to show where the iterates concentrate.

## 2.2 Decreasing step-size

$$x_{n+1} = x_n + a_n(f(x_n) + M_{n+1}),$$

We need the assumptions:

1. $f : \mathbb{R}^d \to \mathbb{R}^d$ is Lipschitz-continuous.
2. The step-sizes $a_n \geq 0$ is such that $\sum_n a_n = +\infty$ and $\sum_n (a_n)^2 = +\infty$.
3. $M_n$ Martingale difference sequence : $\mathbb{E}\left[M_{n+1} | \mathcal{F}_n\right] = 0$ and $\mathbb{E}\left[||M_{n+1}||^2 | \mathcal{F}_n\right] \leq \sigma^2$.
4. $\sup_n \|x_n\|$ remains bounded a.s.

We define $t_n = \sum_{k=0}^{n-1} a_k$ and $\bar{x}$ a piecewise linear function such that $\bar{x}(t(n)) = x_n$. We also write $x_s(t)$ the solution of the ODE $\dot{x} = f(x)$ with $x_s(s) = \bar{x}(s)$.

**Theorem 2.** *For all $T > 0$, we have:*

$$\lim_{s \to \infty} \sup_{t \in [s, s+T]} \|\bar{x}(t) - x_s(t)\| = 0 \textit{ almost surely.}$$

*The sequence $x_n$ converges almost surely to the invariant sets of the ODE $\dot{x} = f(x)$, that is, the set $A$ such that if $x(0) \in A$, then $x(t) \in A$ for all $t > 0$. In particular, if the ODE has a unique attractor $x^*$, then*

$$\lim_{s \to \infty} x_n = x^*.$$

*Proof.* For the first part, we consider $s = 0$ and use the following tools:

1. We compare the ODE and the discrete ODE $y_{n+1} = y_n + a_n f(x_n)$: to show that at $t(n)$: $||y_n - \bar{x}_n|| = O(\sum_k (a_k)^2)$ by Gronwall's inequality.

   Recall the discrete-Gronwall's lemma: if $d_{n+1} = \varepsilon + L\sum_{k=0}^{n} a_k d_k$, then $d_n \leq e^{Lt_n}\varepsilon$ (proof = recurrence + log is convex).

2. Let $B_n = \sum_{k=0}^{L} a_n M_{n+1}$. We have $\text{var}(B_n) \leq \sum_n (a_n)^2\sigma^2$. In particular, $\mathbb{P}(||B_n|| \geq \varepsilon) \leq \sum_n (a_n)^2\sigma^2/\varepsilon^2$ (Chebyshev's inequality). We can extend that to sup by using Doob's inequality and use the supermartingale $B_n^+ = \max_{k\leq b} B_n$?

3. Fix $T$. The idea is now to consider $K_n = \min_{K>n}$ such that $t(K_n) = t_n + T$. By what the assumption on $a_n$, we have $\sum_{k=1}^{K_n} (a_k)^2 \to 0$.

   Similar to our way of defining $y_n$, we can define a $y_{k,n}$ that starts at $x_k$ when $n = k$. Let $m(k)$ be such that $\sum_{\ell=k}^{m(k)} \approx T$. We can show that:

   $$\left\|y_{k,k+m(k)} - x_k + m(k)\right\| \leq e^{LT}\varepsilon,$$

   with probability at least $\sum_{\ell=k}^{m(k)} (a_\ell)^2\sigma^2/\varepsilon^2 < \sum_{\ell=k}^{\infty} (a_\ell)^2\sigma^2/\varepsilon^2$.

   This probability converges to 0 because $\sum_{\ell=1}^{\infty} (a_\ell)^2 < \infty$.

For $t = +\infty$, we write $A = \cap_{t\geq 0}\overline{\cup_{s\geq t}\{\bar{x}(t)\}}$. If should be clear that $x_n \to A$ a.s. $A$ is invariant by using the first part of the lemma and the fact that the flow is invariant. □

Note: we can say more ($A$ is chain transitive).

### 2.2.1 Application to $Q$-learning

For $Q$-learning, we can rewrite the ODE in vector form as:

$$\dot{Q}_{s,a} = \underbrace{r_{s,a} + \gamma \sum_{s'} p(s'|s,a) \max_{a'\in\mathcal{A}} Q_{s',a'} - Q_{s,a}}_{=:f_{s,a}(Q)}$$

The ODE is $\dot{Q} = f(Q)$, the variable is $Q$.

We can verify that this satisfy all assumption for the finite case:

- $f$ is Lipschitz-continuous (because max is.)
- Moreover, the noise is *i.i.d.* if
-
  - If we apply to "synchronous" Q-learning (for all state $s, a$); or
  - If we apply to "asynchronous" Q-learning with a *generative model* (we pick one $(s_t, a_t)$) at random each time.

  If we want to treat the general case, the problem is that the noise is not *i.i.d.*. In this case, we need to treat that we have a "Markovian" noise. This is out of scope of this course.

For $T = +\infty$, we have:

- $f$ can be written as $f(Q) = F(Q) - Q$. We know that $F$ is contrating or the $\|\|_\infty$ (see first course on MDP). Hence, it has a unique fixed point $Q^*$.
- Proving that the ODE converges to $Q^*$ is more complicated. For that, let us denote $u(t) = Q(t) - Q^*$ and assume for now that $F$ is $\alpha$-contracting for the $L_p$ norm. We have:

$$
\begin{aligned}
&\frac{d}{dt}\|u(t)\| \\
&= \frac{d}{dt}(\sum_i |u_i|^p)1/p \\
&= \frac{1}{p}(\sum_i |u_i|^p)1/p - 1\frac{d}{dt}(\sum_i |u_i|^p) \\
&= \|u\|^{1-p}\sum_i \mathrm{sgn}(u_i)|u_i|^{p-1}(F(Q) - Q). \\
&= \|u\|^{1-p}\left[\sum_i \mathrm{sgn}(u_i)|u_i|^{p-1}(F_i(Q) - F_i(Q^*)) - \underbrace{\sum_i \mathrm{sgn}(u_i)|u_i|^{p-1}\underbrace{(Q_i - Q_i^*)}_{=u_i}}_{=\|u\|^p}\right]
\end{aligned}
$$

Recall Hölder: if $1/p + 1/q = 1$, *i.e.*, $q = p/(p-1)$, we have:

$$\sum_i x_i y_i \leq (\sum_i |x_i|^p)^{1/p}(\sum_i |y_i|^q)^{1/q}.$$

Using this with $x_i = F_i(Q) - F_i(Q^*)$ and $y_i = \mathrm{sgn}(u_i)|u_i|^{p-1}$, the first term is smaller than:

$$
\begin{aligned}
\|F(Q) - F(Q^*)\|_p (\sum_i (|u_i|^{p-1})^{p/(p-1)})^{(p-1)/p} &= \|F(Q) - F(Q^*)\|_p \|Q - Q^*\|^{p-1} \\
&\leq \alpha \|Q - Q^*\|_p^p \\
&= \alpha \|u\|^p
\end{aligned}
$$

This shows that $\frac{d}{dt}\|u(t)\| \leq (\alpha - 1)\|u(t)\|$.

The proof for $p = +\infty$ comes by continuity of the norm.

## 2.3 Going further

### 2.3.1 Fluctuations and averaging

Let us go back to $x_{n+1} = x_n + a_n(f(x_n) + M_{n+1})$ and we assume in addition that:

- $\mathbb{E}\left[M_{n+1}M_{n+1}^T|\mathcal{F}_n\right] = Q(x_n)$
- $f$ is twice differentiable.
- The ODE has a unique fixed point that is exponentially stable.

The main idea is to use *generators*. For $n \geq k$, let $y_{k,n}$ be the hybrid term:

$$y_{k,k} = x_k$$
$$y_{k,n+1} = y_{k,n} + a_n f(y_{k,n}).$$

We have:

$$x_n - y_k = y_{n,n} - y_{0,n}$$
$$= \sum_{k=0}^{n-1} y_{k+1,n} - y_{k,n}.$$

Hence, if we can bound $y_{k+1,n} - y_{k,n}$, we are "done".

We can do that by showing that the function $x_k \mapsto y_{k,n}$ is smooth.

This can be used to show variance of order $O(1/n)$ when using $a_n = 1/(n+1)$.

We can do acceleration via averaging. Polyak & Juditsky 92.

### 2.3.2 Constant step-size

Most of the results above also work for the constant step-size, in which case we can show that if there is a unique attractor of the ODE $x^*$, and we use $a = \alpha$, then:

$$\lim_{\alpha \to 0} \lim_{n \to \infty} \mathbb{P}(\text{dist}(x_n^{(\alpha)}) - x^*) = 0$$

We can also obtain fluctuation results. In particular, if the function $f$ is smooth, we get:

$$\lim_{n \to \infty} \mathbb{E}\left[x_n^{(\alpha)}\right] = x^* + C\alpha + O(\alpha^2),$$

but the constant $C$ is a non-zero.

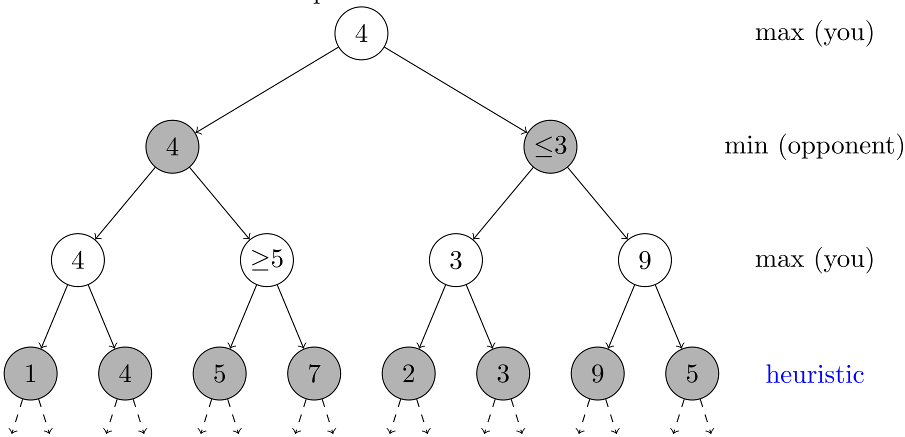
# 3 Monte-Carlon Tree Search

### 3.0.1 Turn-based two players zero sum games

From a given position, takes the best decision. To do so, one can generate a tree of possibilities and explore this tree (*e.g.*), min-max algorithm. But: what if the tree is too big?

## 3.1 Min-max and alpha-beta pruning

You can construct the tree of possibilities




If the tree is two big, you stop at depth $D$ and use a heuristic.

- You can backtrack with the min-max algorithm.
- For optimization, you can use alpha-beta pruning.

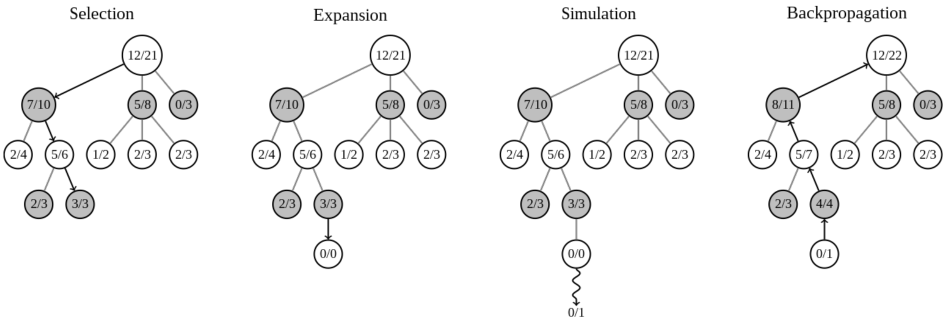
## 3.2 MCTS and exploration

### 3.2.1 Motivation for MCTS

Min-max and alpha-beta perform well (ex: Chess)... but can be limited (ex: go).

- Tree can still be very big ($A^D$)
- You need a good heuristic.
  - Result is only available at the end
- You might want to avoid the exploration of not promising parts.
  - For that you need a good heuristic.

### 3.2.2 MCTS algorithm




(figure from wikipedia)

The algorithm:

- Creates one or multiple children of the leaf.
- Obtains a value of the node (e.g. rollout)
- Backpropagates to the root

For the exploration, one typically uses bandit-like formulas: For each child, let $S(c)$ be the number of success and $N(c)$ be the number of time you played $c$, and $t = \sum_{c'} N(c')$.

- Explore $\arg\max_c \frac{S(c)}{N(c)} + 2\sqrt{\frac{\log t}{N(c)}}$.

Open question: no guarantee with $\sqrt{\log t/N(c)}$. Is $\sqrt{t}/N(c)$ better?

```
while Some time is left do
    Select a leaf node                                       #UCB-like
    Expand a leaf
    Use rollout (or equivalent) to estimate the leaf         #random sampling
    Backpropagate to the root
end while
Return arg max_{c∈children(root)} N(c)                       #or S(c)/N(c).
```

### 3.2.3 Demo / exercice

See the file `connect4.tar.gz` on the website.

# References

[1] Vivek S Borkar. *Stochastic approximation: a dynamical systems viewpoint*, volume 48. Springer, 2009.

[2] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.