Sequential Decision Making

Lecture 4.5 : Summary of the first four courses

Rémy Degenne
(remy.degenne@inria.fr)

Centrale Lille, 2022/2023

Evaluation

TP : the second practical session (January 3rd) will be evaluated.

Project : reading, understanding and explaining a research paper

- Explain the context. You'll need to read parts of one or two of the references.
- Explain the goal of the paper.
- Present the contributions and the limitations of the method. Be critical.
- If possible, implement the algorithm described yourself and report your own findings.

Outcome : a short report (approximately 8 pages).
Good project : relate the paper to others, present your appreciation of the method, evaluate the contributions critically (possibly by doing your own experiments).
Bad project : Rephrase the paper or worse, copy-paste it. Don't do that.

Evaluation

Project : reading, understanding and explaining a research paper

Outcome : a short report (approximately 8 pages).
Good project : relate the paper to others, present your appreciation of the method, evaluate the contributions critically (possibly by doing your own experiments).
Bad project : Rephrase the paper or worse, copy-paste it. Don't do that.

- Up to two students per project
- Deadline for choosing a project : December 16.
- Deadline : January 26, 11 :59pm.
- Send it by email to `remy.degenne@inria.fr`. If you don't have a reply by January 27, noon, send it again.

More details on the website.

Markov Decision Process

A MDP is parameterized by a tuple $(\mathcal{S}, \mathcal{A}, R, P)$ where

- $\mathcal{S}$ is the state space
- $\mathcal{A}$ is the action space
- $R = (\nu_{(s,a)})_{(s,a)\in\mathcal{S}\times\mathcal{A}}$ where $\nu_{(s,a)} \in \Delta(\mathbb{R})$ is the reward distribution for the state-action pair (s, a)
- $P = (p(\cdot|s, a))_{(s,a)\in\mathcal{S}\times\mathcal{A}}$ where $p(\cdot|s, a) \in \Delta(\mathcal{S})$ is the transition kernel associated to the state-action pair (s, a)

In each (discrete) decision time $t = 1, 2, \ldots,$ a learning agent

- selects an action a_t based on his current state s_t (or possibly all the previous observations),
- gets a reward $r_t \sim \nu_{(s_t,a_t)}$
- makes a transition to a new state $s_{t+1} \sim p(\cdot|s_t, a_t)$

[Bellman 1957, Howard 1960, Blackwell 70s...]

Markov Decision Process

A MDP is parameterized by a tuple $(\mathcal{S}, \mathcal{A}, R, P)$ where

- $\mathcal{S}$ is the state space
- $\mathcal{A}$ is the action space
- $R = (\nu_{(s,a)})_{(s,a)\in\mathcal{S}\times\mathcal{A}}$ where $\nu_{(s,a)} \in \Delta(\mathbb{R})$ is the reward distribution for the state-action pair (s, a)
- $P = (p(\cdot|s,a))_{(s,a)\in\mathcal{S}\times\mathcal{A}}$ where $p(\cdot|s,a) \in \Delta(\mathcal{S})$ is the transition kernel associated to the state-action pair (s, a)

Goal : (made more precise later) select actions so as to maximize some notion of *expected* cumulated rewards

Mean reward of action a in state s

$$r(s,a) = \mathbb{E}_{R\sim\nu_{(s,a)}}[R]$$

Different Markov Decision Problems

Overall goal : learn the optimal policy $\pi^\star$ associated to some MDP parameterized by $r(s, a)$ and $p(\cdot|s, a)$ for $(s, a) \in \mathcal{S} \times \mathcal{A}$.

Different contexts :

1. Small state space $\mathcal{S}$, known dynamics
2. Small state space $\mathcal{S}$, unknown dynamics
3. Large state space $\mathcal{S}$, known dynamics
4. Large state space $\mathcal{S}$, unknown dynamics

Value and policy

Value of a policy :

- $V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \middle| s_1 = s\right]$ and $Q^\pi(s,a) = \mathbb{E}^\pi\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \middle| s_1 = s, a_1 = a\right]$.
- $V^*(s) = V^{\pi^*}(s) = \max_\pi V^\pi(s)$ and $Q^*(s,a) = Q^{\pi^*}(s,a) = \max_\pi Q^\pi(s,a)$.
- $V^\pi = \mathbb{E}_{a\sim\pi(s)}[Q^\pi(s,a)]$.

Greedy policy :

- $\text{greedy}(V) = \operatorname{argmax}_{a\in\mathcal{A}}\left(r(s,a) + \gamma\mathbb{E}_{s'\sim p(\cdot|s,a)}\left[V(s')\right]\right)$
- $\text{greedy}(Q) = \operatorname{argmax}_{a\in\mathcal{A}} Q(s,a)$
- $\pi^* = \text{greedy}(V^*)$ and $\pi^* = \text{greedy}(Q^*)$.

Bellman equations and operators

The value of a policy satisfies a **Bellman equation**, written with the **Bellman operator**

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_{a\sim\pi(s)}\left[r(s,a) + \gamma \mathbb{E}_{s'\sim p(\cdot|s,a)}\left[V(s')\right]\right] \ , \\ V^\pi &= T^\pi V^\pi \ . \end{aligned}$$

Similar equations and operators for Q^π, V^*, Q^*.

Properties of V^π and T^π :

- T^π is a γ-contraction
- V^π is the unique fixed point
- $V_{n+1} = T^\pi V_n$ tends to V^π.

Similar properties for Q^π, V^*, Q^*.

Goals

Policy evaluation

Given a policy π, return V^π (or Q^π)

Often : find a "good enough" approximation of V^π.

Finding the best policy

Find $\pi^* = \mathrm{argmax}_\pi V^\pi = \mathrm{argmax}_\pi Q^\pi$.

Property : there exists a deterministic π^*, given by $\pi^* = \mathrm{greedy}(V^*)$ and $\pi^* = \mathrm{greedy}(Q^*)$.
Often : find a policy π which is "close enough" to π^*.

Small MDP, known dynamics

Solve the Bellman equation for policy evaluation : $V^\pi = (\mathbf{I} - \gamma \mathbf{P})^{-1}\mathbf{r}$.

Value iteration for policy evaluation or finding the best policy :

1. Iterate $V_{n+1} = T^\pi V_n$ (resp. $V_{n+1} = T^* V_n$)
2. Stop when $\|V_{n+1} - T^\pi V_n\|$ is small

Then if we are iterating with T^* to find the best policy : return $\pi = \text{greedy}(V_n)$.

Policy iteration for finding the best policy :

1. Use policy evaluation to find V^{π_n}
2. Perform policy improvement : $\pi_{n+1} = \text{greedy}(V^{\pi_n})$.

Both can also be performed with Q instead of V. (Advantages ? Drawbacks ?)

Small MDP, unknown dynamics

Main ideas : Robbins-Monro estimation and temporal differences.

TD(0) for policy evaluation

- $\hat{V}(s_k) \leftarrow \hat{V}(s_k) + \alpha_{N(s_k)}(s_k)\delta(s_k)$ where $\delta(s_k) = r_k + \gamma \hat{V}(s_{k+1}) - \hat{V}(s_k)$ and $(r_k, s_{k+1}) = \mathrm{step}(s_k, \pi)$.

Parallel Robbins-Monro on each state. $\hat{V}$ converges to V^π (under suitable conditions on α, etc.).

Q-Learning for finding the best policy

- $Q(s,a) \leftarrow Q(s,a) + \alpha_{N(s,a)}(s,a)\left(r + \gamma \max_b Q(s', b) - Q(s,a)\right)$ where $(r, s') = \mathrm{step}(s,a)$
- Return $Q, \pi = \mathtt{greedy}(Q)$.

Parallel Robbins-Monro on each state-action pair. Q converges to Q^*. Works for any behaviour policy, provided it explores enough.

Both are modified value iteration / policy iteration, with R-M and TD techniques to deal with unknown dynamics.

Large MDP

Function approximation. Since $\mathcal{F}(\mathcal{S}, \mathbb{R})$ is too large, introduce a (parametric) set of functions $\mathcal{F}_V$ and look for best V in $\mathcal{F}_V$.
Ex : functions representable by a given neural network.

Policy evaluation

- Minimize $\mathtt{MSVE}_\nu(V) = \mathbb{E}_{s\sim\nu}\left[\left(V^\pi(s) - V(s)\right)^2\right]$.
- Use TD(0) semi-gradient. Converges to θ_{TD}
- Or : estimate the solution directly with LSTD, using that $A\theta_{TD} = b$ for some A, b (linear approximation). Variant for Q : LSTD-Q.

Finding the best policy

- LSPI : policy iteration using LSTD-Q for policy evaluation
- Fitted Q-iteration : value iteration for Q, with regression to estimate T^*Q from samples
- Approximate Q-learning : use semi-gradient updates for Q.

And more to come, not value-based.