RLChina 2020

Lecture 2: Foundations of Reinforcement Learning

— from policy methods to PAC bounds analysis

Prof. Jun Wang
UCL

July 28, 2020

Table of content

Main references

- machine learning and learning theory books[1][2]
- reinforcement learning books[3][4]
- approximate dynamic programming [4][5]
- this slide is adopted from our upcoming book chapter[6]

---

[1] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[2] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[3] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[4] Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[5] Rémi Munos. *Introduction to Reinforcement Learning and multi-armed bandits*. NETADIS Summer School, 2013.

[6] Shuang Wu and Jun Wang. *Decision making and AI: a white paper*. 2020.

Table of content

Markov Decision Process(MDP)

Definition:

- an MDP M is a tuple $\{\mathbb{S}, \mathbb{A}, \mathtt{Pr}(s'|s,a), r(s,a,s')\}$
- $\mathbb{S}$: state space; a state $s \in \mathbb{S}$
- $\mathbb{A}$: action space; an action $a \in \mathbb{A}$
- system dynamics: $\mathtt{Pr}(s'|s,a)$
- reward: $r(s,a,s')$ [7]

[7] For the MDP with known state transition $\mathtt{Pr}(s'|s,a)$, the stochastic reward $r(s,a,s')$ can be reduced to a deterministic one as

$$R(s,a) := \mathbb{E}_{s' \sim \mathtt{Pr}(s'|s,a)}[r(s,a,s')].$$

Policy

- a policy specifies what an agent should do in a specific circumstance
- it is a mapping from the history to a (deterministically or randomly) action at present: $\pi_k : (s_0, a_0, s_1, a_1, \ldots, s_k) \mapsto a_k$
  - *Markovian policy* $\pi_k : s_k \mapsto a_k$
  - *stationary* policy $\pi_k = \pi_{k+1}$ for any k
  - *deterministic* policy: $a := \pi(s)$
  - *randomized policy* $\mathtt{Pr}(a|s) := \pi(s|a)$ is often considered for an RL setup as it helps explore the environments

Objectives

an agent chooses a policy in order to maximise one of the following possible objectives:

1. total reward MDP over a finite horizon,

$$J(\pi) := \mathbb{E}\left[\sum_{k=0}^{N} R(s_k, \pi(s_k))\right];$$

2. discounted reward MDP over an infinite horizon,

$$J(\pi) := \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(s_k, \pi(s_k))\right], \text{ for } 0 < \gamma < 1;$$

3. average reward MDP over an infinite horizon (ergodic reward),

$$J(\pi) = \lim_{K \to \infty} \mathbb{E}\left[\frac{1}{K+1} \sum_{k=0}^{K} R(s, \pi(s))\right]$$

Policy evaluation and optimisation

given the objective, the two core questions in an MDP are:

1. how good is a policy? $\implies$ **policy evaluation**
2. what is the optimal policy? $\implies$ **policy optimisation**

Value function and Q function

1. **policy evaluation** *value function*: the expected discounted rewards under a policy π starting from state s,

$$V^\pi(s) := \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(s_k, \pi(s_k)) | s_0 = s\right],$$

and the corresponding action-value function (Q function), the expected reward of taking a particular action, under a policy

$$Q^\pi(s) := R(s, \pi(s)) + \mathbb{E}_{s' \sim \mathtt{Pr}(s'|s,\pi(s))}[V^\star(s')].$$

2. **policy optimisation** the value function optimising the policy gives the optimal value function,

$$V^\star(s) := \max_\pi \mathbb{E}\left[\sum_{k=0}^{\infty} R(s_k, \pi(s_k)) | s_0 = s\right].$$

By the same token, the optimal Q function is

$$Q^\star(s) := max_a R(s, a) + \mathbb{E}_{s' \sim \mathtt{Pr}(s'|s,a)}[V^\pi(s')].$$

Solving the value function and Q function

- **model-based approaches** when the state transition and the reward function are known and given, the MDP can be solved by dynamic programming (DP) [Bel57], e.g., value iteration or policy iteration
- **model-free approaches** if the state transition and the reward function are not given, one can solve MDP with *reinforcement learning* (RL) by *learning* the solution through interactions with the environment

Bellman equation: how good a policy is

evaluation is also known as prediction. by splitting the immediate reward from $V^\pi(s)$ as

$$
\begin{aligned}
V^\pi(s) =& \mathbb{E}\left[\gamma^0 R(s_0, \pi(s_0)) | s_0 = s\right] + \mathbb{E}\left[\sum_{k=1}^{\infty} \gamma^k R(s_k, \pi(s_k)) | s_0 = s\right] \\
=& R(s, \pi(s)) + \mathbb{E}_{s' \sim \mathtt{Pr}(s'|s,\pi(s))}\left[\sum_{k=1}^{\infty} \gamma^k R(s_k, \pi(s_k)) | s_1 = s'\right] \\
=& R(s, \pi(s)) + \gamma \sum_{s'} \mathtt{Pr}(s'|s, a) \underbrace{\mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(s_k, \pi(s_k)) | s_0 = s'\right]}_{=V^\pi(s')}
\end{aligned}
$$

we can obtain the *Bellman equation* for policy π as

$$
V^\pi(s) = R(s, \pi(s)) + \underbrace{\gamma \sum_{s'} \mathtt{Pr}(s'|s, \pi(s)) V^\pi(s')}_{\text{cost}-\text{to}-\text{go}}. \tag{1}
$$

Value iteration for evaluating a policy

when $\mathtt{Pr}(s'|s, a)$ and $R(s, a)$ are given, V^{π} can be computed with either value iteration sketched in Algorithm 1

Algorithm 1 value iteration for evaluating a policy

1: initialise π and V arbitrarily
2: **repeat**
3: $\quad V(s) \leftarrow R(s, \pi(s)) + \gamma \sum_{s'} \mathtt{Pr}(s'|s, \pi(s)) V(s'), \ \forall s$
4: **until** convergence

both π and V functions can be as a form of tables (tabular).

Bellman optimality equation: optimal policy

optimisation is also called as control. by splitting the reward, we can obtain the *Bellman optimality equation* as

$$V^\star(s) = \max_{a \in \mathbb{A}} R(s,a) + \underbrace{\gamma \sum_{s'} \mathtt{Pr}(s'|s,a) V^\star(s')}_{\text{cost}-\text{to}-\text{go}} . \tag{2}$$

when $\mathtt{Pr}(s'|s,a)$ and $R(s,a)$ are given, the optimal value function and the optimal policy can be directly solved with value iteration [Bel57] and policy iteration [How60], respectively.

$$\textbf{(value iteration)} \quad V(s) \leftarrow \max_{a \in \mathbb{A}} R(s,a) + \gamma \sum_{s'} \mathtt{Pr}(s'|s,a) V(s') \tag{3}$$

$$\textbf{(policy iteration)} \quad \pi(s) \leftarrow \arg\max_{a \in \mathbb{A}} R(s,a) + \gamma \sum_{s'} \mathtt{Pr}(s'|s,a) V^\pi(s') \tag{4}$$

VI

Algorithm 2 Value iteration for optimising a policy

1: Initialize π and V arbitrarily
2: **repeat**
3: $\quad V(s) \leftarrow \max_{a \in \mathbb{A}} R(s,a) + \gamma \sum_{s'} \mathtt{Pr}(s'|s,a) V(s'),\ \forall s$ $\triangleright$ value improvement
4: **until** convergence

- value iteration computes the optimal value function $V^\star$ under the optimal policy
- using $V^\star(s)$, the optimal policy $\pi^\star$ can be calculated from the Bellman optimality equation by

$$a^\star = \pi^\star(s) = \arg\max_{a \in \mathbb{A}} R(s,a) + \gamma \sum_{s'} \mathtt{Pr}(s'|s,a) V^\star(s').$$

PI

Algorithm 3 Policy iteration for optimising a policy

Initialize π and V arbitrarily
repeat
    repeat
        $V(s) \leftarrow R(s, \pi(s)) + \gamma \sum_{s'} \mathtt{Pr}(s'|s, \pi(s)) V(s'), \ \forall s$ $\triangleright$ Policy evaluation using Algorithm 1
    until Convergence
    $\pi(s) \leftarrow \arg\max_{a \in \mathbb{A}} R(s, a) + \gamma \sum_{s'} \mathtt{Pr}(s'|s, a) V(s'), \ \forall s$ $\triangleright$ Policy improvement
until convergence

- policy iteration directly generates the optimal policy

VI and PI: convergence analysis

Let $\mathbb{F}$ be the space of functions on domain $\mathbb{S}$. Define the Bellman *policy* operator $T^\pi : \mathbb{F} \mapsto \mathbb{F}$ and the Bellman *optimality* operator $T : \mathbb{F} \mapsto \mathbb{F}$ as

$$T^\pi V(s) := R(s,a) + \gamma \sum_{s'} \mathtt{Pr}(s'|s,a) V(s'), \ \forall s, \tag{5}$$

$$TV(s) := \max_{a \in \mathbb{A}} R(s,a) + \gamma \sum_{s'} \mathtt{Pr}(s'|s,a) V(s') \ \forall s \tag{6}$$

VI and PI: convergence analysis

- these two operators are mappings from one value function $V(s)$ to another $TV(s)$
- also they are both *monotonic* and *contraction* mappings (with respect to the ∞-norm), i.e.,
  1. *monotonicity*, if $V(s) \geq U(s)$ for any s,

$$T^{\pi} V(s) \geq T^{\pi} U(s); \tag{7}$$
$$TV(s) \geq TU(s) \tag{8}$$

  2. *contraction*, for any U, V,

$$\| T^{\pi} V - T^{\pi} U \|_{\infty} \leq \gamma \| V - U \|_{\infty} \tag{9}$$
$$\| TV - TU \|_{\infty} \leq \gamma \| V - U \|_{\infty} \tag{10}$$

Monotonicity proof

the inequality (7) follows from

$$T^\pi V(s) - T^\pi U(s) = \sum_{s'} \mathtt{Pr}(s'|s, \pi(s))(V(s') - U(s')) \geq 0.$$

the inequality (8) follows from, for any a,

$$R(s, a) + \sum_{s'} \mathtt{Pr}(s'|s, a)V(s') \geq R(s, a) + \sum_{s'} \mathtt{Pr}(s'|s, a)U(s').$$

Contraction mapping proof

The inequality (9) follows from

$$\|T^\pi V - T^\pi U\|_\infty = \max_s \gamma \sum_{s'} \texttt{Pr}(s'|s,\pi(s))|V(s') - U(s')|$$

$$\leq \gamma \Big(\sum_{s'} \texttt{Pr}(s'|s,\pi(s)) \Big) \max_{s'} |V(s') - U(s')| \leq \gamma \|U - V\|_\infty .$$

The inequality (10) follows from

$$\begin{aligned}
\|TV - TU\|_\infty =& \max_s | \max_a \{R(s,a) + \gamma \sum_{s'} \texttt{Pr}(s'|s,a)V(s')\} \\
& - \max_a \{R(s,a) + \gamma \sum_{s'} \texttt{Pr}(s'|s,a)U(s')\}| \\
\leq & \max_{s,a} |R(s,a) + \gamma \sum_{s'} \texttt{Pr}(s'|s,a)V(s') \\
& - R(s,a) - \gamma \sum_{s'} \texttt{Pr}(s'|s,a)V(s')| \\
=& \gamma \max_{s,a} |\sum_{s'} \texttt{Pr}(s'|s,a)(V(s') - U(s'))| \\
\leq & \gamma \Big(\sum_{s'} \texttt{Pr}(s'|s,a) \Big) \max_{s'} |(V(s') - U(s'))| \leq \gamma \|V - U\|_\infty .
\end{aligned}$$

Contraction mapping proof

for any contraction operator (take T for example), we have a unique fixed point $V^\star = TV^\star$ by Cauchy convergence theorem[8]. We therefore have the convergence as

$$\begin{aligned} \|V_{k+1} - V^\star\|_\infty &= \|TV_k - TV^\star\|_\infty \\ \leq \gamma\|V_k - V^\star\|_\infty \leq \cdots &\leq \gamma^{k+1}\|V_0 - V^\star\|_\infty \to 0. \end{aligned}$$

8

- a sequence of real numbers (a_n) is said to be a Cauchy Sequence if $\forall\epsilon > 0$ there exists an $N \in \mathbb{N}$ such that if $m, n \geq N$ then $|a_n - a_m| < \epsilon$
- theorem (Cauchy Convergence Criterion): If (a_n) is a sequence of real numbers, then (a_n) is convergent if and only if (a_n) is a Cauchy sequence

Policy iteration: convergence

- policy iteration monotonically improves the policy by

$$
\begin{aligned}
V^{\pi_{k+1}} =& (I - \gamma P^{\pi k+1})^{-1} R^{\pi_{k+1}} \\
\geq& (I - \gamma P^{\pi_{k+1}})^{-1} (V^{\pi_k} - \gamma P^{\pi_{k+1}} V^{\pi_k}) \qquad (11) \\
=& V^{\pi_{\mathrm{old}}},
\end{aligned}
$$

where (11) follows from

$$
\begin{aligned}
R^{\pi_{k+1}} + \gamma P^{\pi_{k+1}} V^{\pi_k} = T^{\pi_{k+1}} V^{\pi_k} = T V^{\pi_k} \geq V^{\pi_k} \\
\iff R^{\pi_{k+1}} \geq (I - \gamma P^{\pi_{k+1}}) V^{\pi_k}.
\end{aligned}
$$

by the monotone convergence theorem[9], $V^{\pi^\star} = \lim_{k\to\infty} V^{\pi_k}$ exists and satisfies

$$
V^{\pi^\star} = T V^{\pi^\star},
$$

which satisfies the Bellman optimality equation.

[9] If a sequence of real numbers is increasing and bounded above, then its supremum is the limit.

Problems with VI and PI

- *continuous states and action*. both VI and PI fail when the states and actions are continuous
- *curse of dimensionality*. both VI and PI involve iterative scheme over the whole state space. The algorithm is not scalable with respect to the size of the state space
- *partial observable states*. in reality, the states may not be fully observable. the partial observability model leads to a partially observable MDP (**POMDP**)
- *unknown model*. using DP to solve an MDP requires knowledge of $\mathtt{Pr}(s'|s, a)$ and $R(s, a)$. These quantities are costly or even impossible to acquire, especially for huge state space and action space.
  - nevertheless, the agent can collect samples of state transitions $s, a \to s'$ and associated reward $r(s, a, s')$ through interaction with the environment.
  - this suits machine learning and motivates the development of reinforcement learning

Q learning

- Q-learning [Wat89] learns from the estimated optimal value function
- it defines action-value function $Q^\pi(s,a)$ for policy π, which reflects the future reward for different actions under π as

$$\begin{aligned} Q^\pi(s,a) &= \sum_{s'} \mathtt{Pr}(s'|s,a)\Big(r(s,a,s') + \gamma V^\pi(s')\Big) \\ &= \sum_{s'} \mathtt{Pr}(s'|s,a)\Big(r(s,a,s') + \gamma Q^\pi(s', \pi(s'))\Big). \end{aligned}$$

- the optimal $Q^\star(s,a)$ is related to the optimal value function $V^\star(s) = \max_a Q^\star(s,a)$, thus

$$Q^\star(s,a) = \sum_{s'} \mathtt{Pr}(s'|s,a)[r(s,a,s') + \gamma \max_{a'} Q(s',a')].$$

- plugging $Q(s,a) = Q^\star(s,a)$ into the value iteration for optimising policy and **replace the expectation with its sample**, we have

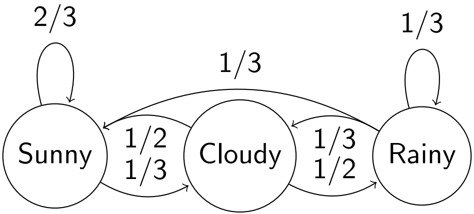
$$(\textbf{Q-learning}) \quad Q(s,a) \leftarrow Q(s,a) + \alpha[\underbrace{r + \gamma \max_{a'} Q(s',a')}_{\text{sample of } Q^\star(s,a)} - Q(s,a)].$$

Roadmap

Table of content

Markov chains: an example

- *sunny*, *rainy*, and *cloudy* are called the states of the Markov chain
- if the weather is currently *sunny*, what is the prediction for the next few days according to the model?



Day	sunny	cloudy	rainy
0	1	0	0
1	2/3	1/3	0
2	0.611	0.222	0.167
3	0.574	0.259	0.167
...	0.568	0.247	0.185

n-step transition probability

- The n-step transition probability of a Markov chain is the probability that it goes from state i to state j in exactly n steps (transitions):

$$p_{ij}^{(n)} := P(S_{n+m} = j | S_m = i)$$

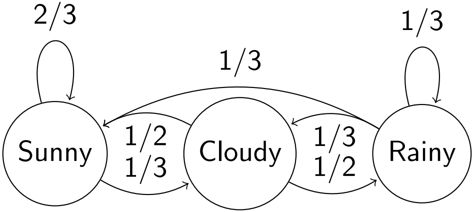
- define $P = (p_{ij})$ the transition matrix; then the n-step transition matrix is given by n powers of the matrix:

$$P^{(n)} = P^n, for\ n \geq 1$$

- $p(\textbf{i to j in n steps})$ = sum of probabilities of all paths i to j in n steps

$$p_{ij}^{(n+m)} = \sum_k p_{ik}^{(m)} p_{kj}^{(n)}$$

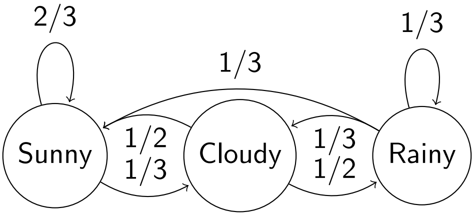
n-step transition probability: an example



$$P^3 \approx \begin{pmatrix} 0.574 & 0.259 & 0.167 \\ 0.556 & 0.222 & 0.222 \\ 0.537 & 0.259 & 0.204 \end{pmatrix} ; P^{10} \approx \begin{pmatrix} 0.563 & 0.250 & 0.187 \\ 0.562 & 0.250 & 0.187 \\ 0.562 & 0.250 & 0.188 \end{pmatrix}$$

- regardless of the initial weather $q^{(1)}$ is, $q^{(1)} \cdot P^n$ seems to approach $\tilde{q} \approx (0.563, 0.250, 0.188)$ as n grows
- if we multiply the vector $\tilde{q}$ with P, we almost get $\tilde{q}$ again, e.g., $\tilde{q}$ is almost an eigenvector of P with eigenvalue 1

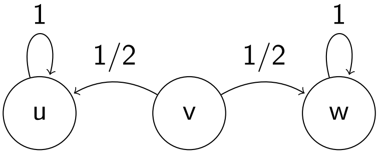
n-step transition probability: an example



$$P^3 \approx \begin{pmatrix} 0.574 & 0.259 & 0.167 \\ 0.556 & 0.222 & 0.222 \\ 0.537 & 0.259 & 0.204 \end{pmatrix}; P^{10} \approx \begin{pmatrix} 0.563 & 0.250 & 0.187 \\ 0.562 & 0.250 & 0.187 \\ 0.562 & 0.250 & 0.188 \end{pmatrix}$$

- a distribution q over the states is *stationary distribution* of the Markov chain with transition matrix P if $q = q \cdot P$
- q could be interpreted as the *long-term visitation rate*

But not all Markov chain has the stationary distribution



- this Markov chain has infinitely many stationary distributions, for example $q = (1, 0, 0)$, $q^* = (0, 0, 1)$, and $q^\star = (0.2, 0, 0.8)$. The states u and w are called absorbing states, since they are never left once they are entered (dead-ends)

Ergodic Markov chains

- even without dead-ends, a graph may not have well-defined long-term visit rates;
  - *requirement* 1: there is a path from any state to any other state
  - *requirement* 2: the states cannot be partitioned such that the random walker visits the partitions sequentially (no loop!)
- in an Ergodic Markov Chains:
  - The $p^{(n)}$ has settled to a limiting value q

$$p_{ij}^{(n)} \to q_j, \text{ as } n \to \infty$$

  - This value is independent of initial state $q^{(1)}$
  - The $q^{(n)}$ also approaches this limiting value: $q_j^{(n)} \to q_j$
  - where q is the unique stationary distribution of the chain (i.e. the limiting distribution is the stationary distribution)

Go back to RL: the objective function

- suppose a policy, denoted as π_θ, is parameterised by θ
- the expected reward (as the objective):

$$J(\theta) = \sum_{s \in S} d^\pi(s) V^\pi(s) = \sum_{s \in S} d^\pi(s) \left(\sum_{a \in A} \pi_\theta(a \mid s) Q^\pi(s, a) \right)$$

  where $d^\pi(s) := \lim_{t \to \infty} p\left(S_t = s \mid s_0, \pi_\theta\right)$ is the stationary distribution of the Markov chain when the agent starts from s_0 and following policy π_θ for t steps

  - When π_θ is given, as the time progresses, the probability that the agent ends up with one state becomes unchanged (regardless of s_0) if the underlying Markov chain is *ergodic*
  - we thus drop s_0 in $J(\theta)$

Policy gradient

the expected reward (as the objective):

$$J(\theta) = \sum_{s \in S} d^{\pi}(s) V^{\pi}(s) = \sum_{s \in S} d^{\pi}(s) \left(\sum_{a \in A} \pi_{\theta}(a \mid s) Q^{\pi}(s, a) \right)$$

where $d^{\pi}(s) := \lim_{t \to \infty} p\left(S_t = s \mid s_0, \pi_{\theta}\right)$

- *gradient ascent* moves θ toward the direction suggested by the gradient $\nabla_{\theta} J(\theta)$ to find the best θ for π_{θ} that produces the highest return
- but, computing $\nabla_{\theta} J(\theta)$ is tricky as it depends on both the action selection (directly determined by π_{θ}) and the stationary distribution of states d^{π} (indirectly determined by π_{θ})
- given that the environment is generally unknown, it is difficult to estimate the effect on the state distribution by a policy update.

Policy gradient theorem

- **Policy gradient theorem**[10]: for an MDP,

$$\nabla_\theta J(\theta) = \sum_{s \in S} d^\pi(s) \left(\sum_{a \in A} Q^\pi(s, a) \nabla_\theta \pi_\theta(a \mid s) \right)$$

- It provides a nice reformation of the objective function that does not involve the derivative of the state distribution $\frac{\partial d^\pi(s)}{\partial \theta}$

[10] Richard S Sutton et al. "Policy gradient methods for reinforcement learning with function approximation". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 1999, pp. 1057–1063.

Policy gradient theorem [SB98]: the proof

$$\nabla_\theta \left(\mathrm{E}\left[\sum_{k=1}^{\alpha} \gamma^{k-1} r_{t+k} \mid s_t = s_0, \pi \right] \right) = \nabla_\theta V^\pi(s_0) = \nabla_\theta \left(\sum_{a \in A} Q^\pi(s_0, a) \pi_\theta(a \mid s_0) \right)$$

$$= \sum_{a \in A} \left(Q^\pi(s_0, a) \nabla_\theta \pi_\theta(a \mid s_0) + \pi_\theta(a \mid s_0) \nabla_\theta Q^\pi(s_0, a) \right) \qquad \textit{product rule}$$

$$= \sum_{a \in A} \left(Q^\pi(s_0, a) \nabla_\theta \pi_\theta(a \mid s_0) + \pi_\theta(a \mid s_0) \nabla_\theta \left(\sum_{s', r} P(s', r \mid s, a) \left(r + V^\pi(s') \right) \right) \right) \qquad \textit{expand}$$

$$= \sum_{a \in A} \left(Q^\pi(s_0, a) \nabla_\theta \pi_\theta(a \mid s_0) + \pi_\theta(a \mid s_0) \left(\sum_{s', r} P(s', r \mid s_0, a) \nabla_\theta V^\pi(s') \right) \right) \qquad \textit{remove } r$$

$$= \sum_{a \in A} \left(Q^\pi(s_0, a) \nabla_\theta \pi_\theta(a \mid s_0) + \pi_\theta(a \mid s_0) \left(\sum_{s'} P(s' \mid s_0, a) \nabla_\theta V^\pi(s') \right) \right) \qquad \textit{marginalise r ou}$$

$$= \sum_{a \in A} \left(Q^\pi(s_0, a) \nabla_\theta \pi_\theta(a \mid s_0) \right) + \sum_{a \in A} \pi_\theta(a \mid s_0) \left(\sum_{s'} P(s' \mid s_0, a) \nabla_\theta V^\pi(s') \right)$$

$$= \sum_{a \in A} \left(Q^\pi(s_0, a) \nabla_\theta \pi_\theta(a \mid s_0) \right) + \sum_{s'} \left(\sum_{a \in A} \pi_\theta(a \mid s_0) P(s' \mid s_0, a) \right) \nabla_\theta V^\pi(s')$$

Policy gradient theorem [SB98]: the proof

- we thus have the following recursive form of the gradient:

$$\nabla_\theta V^\pi(s_0) = \sum_{a \in A} \left(Q^\pi(s_0, a) \nabla_\theta \pi_\theta(a \mid s_0) \right) + \sum_{s\prime} \left(\sum_{\mathrm{a} \in A} \pi_\theta(a \mid s_0) P(s' \mid s_0, a) \right) \nabla_\theta V^\pi(s')$$

- to simplify this, we have:

$$\nabla_\theta V^\pi(s_0) = \varphi(s_0) + \sum_{s'} P^\pi(s' \mid s_0) \nabla_\theta V^\pi(s'),$$

where $\varphi(s_0) := \sum_{a \in A} \left(Q^\pi(s_0, a) \nabla_\theta \pi_\theta(a \mid s_0) \right)$ and Markov transition $P^\pi(s'|s_0) := \sum_{\mathrm{a} \in A} \pi_\theta(a \mid s_0) P(s' \mid s_0, a)$

Policy gradient theorem [SB98]: the proof

- We now consider the following visitation sequence:

$$s_0 \frac{a \sim \pi_\theta(. \mid s_0)}{\longrightarrow} s' \frac{a \sim \pi_\theta(. \mid s')}{\longrightarrow} s'' \frac{a \sim \pi_\theta(\mid s'')}{\longrightarrow} \ldots \frac{a \sim \pi_\theta(. \mid .)}{\longrightarrow} s$$

- and denote the probability of transitioning from state s_0 to state s with policy π_θ after k step as

$$P^\pi\left(s'' \mid s_0, k\right) \equiv \sum_{s'} P^\pi\left(s'' \mid s', k-1\right) P^\pi\left(s' \mid s_0\right)$$

- where we have $P^\pi(s \mid s_0, 0) = 1$ if $s = s_0$ and $P^\pi(s \mid s_0, 0) = 0$ if $s \neq s_0$ (because it happened already!)
- Let us now go back to unroll the gradient of the value function:

$$\nabla_\theta V^\pi(s_0) = \varphi(s_0) + \sum_{S'} P^\pi\left(s' \mid s_0\right) \nabla_\theta V^\pi\left(s'\right)$$

Policy gradient theorem: the proof

let us now go back to unroll the gradient of the value function:

$$\nabla_\theta V^\pi(s_0) = \varphi(s_0) + \sum_{s'} P^\pi(s' \mid s_0) \nabla_\theta V^\pi(s')$$

$$= \varphi(s_0) + \sum_{s'} P^\pi(s' \mid s_0, 1) \left[\varphi(s') + \sum_{s'} P^\pi(s'' \mid s') \nabla_\theta V^\pi(s'') \right]$$

$$= \varphi(s_0) + \left[\sum_{s'} P^\pi(s' \mid s_0, 1) \varphi(s') \right] + \left[\sum_{s'} P^\pi(s' \mid s, 1) \sum_{s''} P^\pi(s'' \mid s') \nabla_\theta V^\pi(s'') \right]$$

$$= \varphi(s_0) + \left[\sum_{s'} P^\pi(s' \mid s_0, 1) \varphi(s') \right] + \left[\sum_{s''} \sum_{s'} P^\pi(s' \mid s, 1) P^\pi(s'' \mid s') \nabla_\theta V^\pi(s'') \right]$$

$$= \varphi(s_0) + \left[\sum_{s'} P^\pi(s' \mid s_0, 1) \varphi(s') \right] + \left[\sum_{s''}^{\infty} P^\pi(s' \mid s, 2) \nabla_\theta V^\pi(s'') \right]$$

$$= \sum_{s \in S} \sum_{k=0}^{\infty} P^\pi(s \mid s_0, k) \varphi(s) = \sum_{s \in S} \sum_{k=0}^{\infty} \left[P^\pi(s \mid s_0, k) \sum_{a \in A} (Q^\pi(s, a) \nabla_\theta \pi_\theta(a \mid s)) \right]$$

Policy gradient theorem: the proof

- if we define $d^{\pi}(s) := \sum_{k=0}^{\alpha} P^{\pi}(s \mid s_0, k)$ as the non-normalised visitation probabilities of state s (starting from s_0), the following

$$\nabla_{\theta} V^{\pi}(s_0) = \sum_{s \in S} \sum_{k=0}^{\alpha} \left[P^{\pi}(s \mid s_0, k) \sum_{a \in A} (Q^{\pi}(s, a) \nabla_{\theta} \pi_{\theta}(a \mid s)) \right]$$

becomes

$$\nabla_{\theta} V^{\pi}(s_0) = \sum_{s \in S} d^{\pi}(s) \sum_{a \in A} (Q^{\pi}(s, a) \nabla_{\theta} \pi_{\theta}(a \mid s))$$

- in the episodic case, $d^{\pi}(s)$ the average length of an episode; in the continuing case it is 1.
- a Markov chain is ergodic $\rightarrow d^{\pi}(s)$ a unique (non-normalised) stationary visiting prob. regardless of s_0

REINFORCE

- in order to make an unbiased estimation, the gradient can be further written as[11]:

$$
\begin{aligned}
\nabla_\theta V^\pi(s_0) &= \sum_{s \in S} d^\pi(s) \sum_{a \in A} \left(Q^\pi(s,a) \nabla_\theta \pi_\theta(a \mid s)\right) \\
&\propto \sum_{s \in S} \mu(s) \sum_{a \in A} \left(\pi_\theta(a \mid s) Q^\pi(s,a) \frac{\nabla_\theta \pi_\theta(a \mid s)}{\pi_\theta(a \mid s)}\right) \\
&= \mathrm{E}_{s \sim \mu, a \sim \pi}\left[Q^\pi(s,a) \nabla_\theta \ln \pi_\theta(a \mid s)\right] \\
&= \mathrm{E}_{s \sim \mu, a \sim \pi}\left[G_t \nabla_\theta \ln \pi_\theta(a \mid s)\right] \\
\text{where } & \mathrm{E}_{s \sim d^\pi, a \sim \pi}\left[G_t \mid s, a\right] = Q^\pi(s,a)
\end{aligned}
$$

- so the gradient update is $\theta_{t+1} = \theta_t + \alpha G_t \nabla_\theta \ln \pi_\theta(a \mid s)$
- it is Monte Carlo Policy Gradient as REINFORCE uses the complete return from time t, which includes all future rewards up until the end for the episode

[11] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3-4 (1992).

PG algorithm

Algorithm 4 Policy gradient with Monte-Carlo simulator

Initialize θ
repeat
    Sample trajectories $\{\tau_i\}$ with horizon H using $\pi_\theta(a|s)$
    $G_{i,t} \leftarrow \sum_{t=t'}^{H} \gamma^{t-t'} R(s_{i,t}, a_{i,t})$
    $V_t \leftarrow \frac{1}{M} \sum_{i=1}^{M} G_{i,t}$
    $A(s_{i,t}, a_{i,t}) \leftarrow G_{i,t} - V_t$
    $\Delta \leftarrow \sum_{i,t} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) A(s_{i,t}, a_{i,t})$
    $\theta \leftarrow \theta + \alpha\Delta$
until convergence

typically replace $Q^{\pi_\theta}(s, a)$ with advantage function

$$A^{\pi_\theta}(s, a) := Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s).$$

in order to reduce the variance of the gradient

Table of content

Probably Approximately Correct (PAC) learning

- *learnability* is a key concept in ML:
  - what concepts can be learned?
  - how efficient is a particular learning method?
  - what is inherently hard to learn?
  - how many examples are needed in order to learn a concept successfully?
  - is there any generic model and theory about learnability?
- the Probably Approximately Correct (PAC) learning framework [Val84] is designed to answer the following two critical questions:
  1. sample complexity — how many training examples do we need to converge to a successful hypothesis with a high probability?
  2. computational complexity — how much computational effort is needed to converge to a successful hypothesis with a high probability?

PAC: definition and notation

- X as the set of all possible instances or examples, e.g., the set of images containing faces or non-faces classes
- Y is the concept class, a set of target concepts y, e.g., face, non-face
- consider $f : X \rightarrow \ Y = \{0,1\}$ the target concept to learn
- define D the target distribution, a fixed probability distribution over X. We shall make sure that the training and test examples are drawn according to D
- define a set of training samples as S and a set of concept hypotheses H, e.g., the set of all linear classifiers

the learning problem is to, given a limited set of sample S, learn a hypothesis $h : X \rightarrow \mathrm{Y} \in H$ that approximating f. note that f may be in H or may not.

PAC: definition and notation

- we then define two types of errors in order to understand the approximation
- true error or generalisation error of h is given as $R(h) = \Pr_{x \sim D}[h(x) \neq f(x)] = E_{x \sim D}\left[1_{h(x) \neq f(x)}\right]$,
- whereas the average error of h on the training sample S is given according to the empirical distribution $\hat{D}$ for the set S: $\hat{R}_S(h) = \Pr_{x \sim \hat{D}}[h(x) \neq f(x)] = E_{x \sim \hat{D}}\left[1_{h(x) \neq f(x)}\right] = \frac{1}{m}\sum_{i=1}^{m} h(x_i) \neq f(x_i)$.
- note that $R(h) = E_{S \sim D^m}\left[\hat{R}_S(h)\right]$, where D^m is a distribution of sampling S according to D.

PAC

Definition PAC Learning [Val84]: A concept class Y is PAC-learnable if there exists an algorithm $L(S) \to h$ such that:

- for all $y \in Y$ and $\delta > 0$ and $\varepsilon > 0$ and all distributions D,

$$\Pr_{S \sim D^m}\left[R\left(h\right) \leq \varepsilon\right] \geq 1 - \delta$$

- for samples S of size $m \geq$ poly($1/\varepsilon$, $1/\delta$), where poly() is a polynomial function.

PAC makes use of $\delta > 0$ to define the confidence $1-\ \delta$ (*probabilistically*) and $\varepsilon > 0$ the accuracy 1 – ε (*approximate correct*)
A concept class Y is thus PAC-learnable if the hypothesis returned by the algorithm after observing a number of points polynomial in $1/\varepsilon$ and $1/\delta$ is approximately correct (error at most ε) with high probability (at least $1\ -\ \delta$)

Learning bound for finite H - consistent case

- **theorem**: let H be a finite set of functions from X to $\{0,1\}$ and L an algorithm that for any target concept $y \in Y$ and sample S returns a *consistent*[12] hypothesis $h_S : R_S(h_S) = 0$. then, for any $\delta > 0$, with probability at least $1-\delta$

$$R(h_S) \leq \underbrace{\frac{1}{m}(\log|H| + log\frac{1}{\delta})}_{\epsilon} \qquad \textit{generalisation bound}$$

  the upper bound increases with $\log|H|$ or the related term $\log_2|H|$, which can be interpreted as the number of bits needed to represent H

- Equivalently $P_{S\ D^m}[R(h_S) \leq \epsilon] \geq 1-\delta$ holds if

$$m \geq \frac{1}{\epsilon}(\log|H| + \log\frac{1}{\delta})$$

[12] a hypothesis set is consistent if it admits no error on training sample S

Learning bound for finite H - consistent case

- **proof**: for any $\epsilon > 0$, we define $H_\epsilon = \{h \in H : R(h) > \epsilon\}$
  We then have:

$$
\begin{aligned}
& Pr[\exists h \in H_\epsilon : \hat{R}_S(h) = 0] && \Leftarrow R > \epsilon \cup R_S(h) = 0 \\
=& Pr[\hat{R}_S(h_1) = 0 \cup \ldots \cup \hat{R}_S(h_{|H_\epsilon|}) = 0] && \\
\leq& \Sigma_{h \in H_\epsilon} Pr[R_S(h) = 0] && \Leftarrow \textit{union bound} \\
\leq& \Sigma_{h \in H_\epsilon} (1-\epsilon)^m && \Leftarrow \text{no error in any } m \text{ samples} \\
\leq& |H|(1-\epsilon)^m && \Leftarrow |H_\epsilon| \leq |H| \\
\leq& |H| e^{-m\epsilon} && \Leftarrow 1 - \epsilon \leq e^{-\epsilon}
\end{aligned}
$$

Learning bound for finite H: inconsistent Case

- no $h \in H$ is a consistent hypothesis
- the typical case in practice: difficult problems, complex concept class
- but, inconsistent hypotheses with a small number of errors on the training set can be useful
- need a more powerful tool: Hoeffding's inequality

Hoeffding's inequality

- **corollary**: for any $\epsilon > 0$ and any hypothesis $h : X \to \{0, 1\}$ the following inequalities holds:

$$Pr[|R(h) - \hat{R}(h)| \geq \epsilon] \leq 2e^{-2m\epsilon^2}$$

- this is due to:

$$Pr[R(h) - \hat{R}(h) \geq \epsilon] \leq e^{-2m\epsilon^2}$$

$$Pr[\hat{R}(h) - R(h) \geq \epsilon] \leq e^{-2m\epsilon^2}$$

- proof can be derived directly from Hoeffding's inequality [MRT18]

Learning bound for finite H: inconsistent Case

- **theorem**: H is a finite hypothesis set. for any $\delta > 0$, with probability at least $1 - \delta$,

$$\forall h \in H, R(h) \leq \hat{R}_S(h) + \sqrt{\frac{log|H| + log\frac{2}{\delta}}{2m}}$$

- **proof**: By the union bound, we have

$$\begin{aligned}
&Pr[max_{h\in H}|R(h) - \hat{R}_S(h)| \geq \epsilon] \\
=&Pr[|R(h_1) - \hat{R}_S(h_1)| \geq \epsilon \cup ... \cup |R(h_|H|) - \hat{R}_S(h_|H|)| \geq \epsilon] \\
\leq&\Sigma_{h\in H}Pr[|R(h) - \hat{R}_S(h)| \geq \epsilon] \qquad \Leftarrow \textit{union bound} \\
\leq&2|H|e^{-2m\epsilon^2} \qquad \Leftarrow \textit{apply previous corollary}
\end{aligned}$$

Estimation and approximation errors

- H is a set of functions mapping from X to $\{0,1\}$. The *excess error* (generalised error against Bayesian error[13] $R^\star$) of a $h \in H$ can be decomposed as follows:

$$R(h_S) - R^\star = \underbrace{\left(R(h_S) - \inf_{h\in H} R(h)\right)}_{estimation} + \underbrace{\left(\inf_{h\in H} R(h) - R^\star\right)}_{approximation}$$

  - *estimation error:* it depends on the hypothesis h_S selected. it measures the error due to the samples
  - *approximation error:* it measures how well the Bayes error can be approximated using H

[13] The infimum of the errors achieved by any measurable functions: $\min\{P[0|x], P[1|x]\}$

Table of content

Approximate methods

- when the state space is huge, approximate the value function by (parameterised) function approximators, e.g., linear model or neural networks
- In general, ADP (Approximate DP) approximates the optimal value function $V^\star$ by finding V in some function space $\mathcal{V}$ as

$$V = \arg\min_{U \in \mathcal{V}} \text{distance}(V^\star, U)$$

  with some distance metric and use the approximator V to generate a greedy policy as

$$\pi(s) = \arg\max_a R(s,a) + \gamma \sum_{s'} \mathtt{Pr}(s'|s,a) V(s).$$

  - Most modern RL algorithms can be viewed as solving ADP by sampling the transition $\mathtt{Pr}(s'|s,a)$ and rewards $r(s,a,s')$
  - Unlike DP, ADP algorithms may not converge as the composition of Bellman operator T and projection onto the function space $\mathcal{V}$ is not a contraction general, resulting in oscillating and even divergent behavior like Q-learning
  - Nevertheless, we can analyze its **performance bounds** and **sample complexity bounds**

ADP performance bounds

- DP is guaranteed to converge and consequently leads to convergence of RL with tabular MDP
- Unlike DP, not all ADP algorithms converge and ADP may not find the actual (optimal) value functions for $V^\star$ is not necessarily in $\mathcal{V}$
- The performance gap of a policy π generated from V and the actual optimal policy $\pi^\star$ has certain theoretic upper bounds
- One is given by difference between the optimal value function and the approximator V itself

$$\| \underbrace{V^\star - V^\pi}_{\text{performance gap}} \|_\infty \leq \frac{2\gamma}{1-\gamma} \| \underbrace{V^\star - V}_{\text{approximation error}} \|_\infty .$$

This follows from

$$\begin{aligned}
\|V^\star - V^\pi\|_\infty \leq & \|V^\star - T^\pi V\|_\infty + \|T^\pi V - T^\pi V^\pi\|_\infty \\
\leq & \|TV^\star - TV\|_\infty + \gamma \|V - V^\pi\|_\infty \\
\leq & \gamma \|V^\star - V\|_\infty + \gamma(\|V - V^\star\|_\infty + \|V^\star - V^\pi\|_\infty) \\
\leq & \frac{2\gamma}{1-\gamma} \|V^\star - V\|_\infty . \qquad (12)
\end{aligned}$$

AVI and API

- this approximation error defined bound motivates algorithms such as AVI and API to minimise the approximation error
- formally, the approximate value iteration (AVI) is written as

$$(\textbf{AVI}) \quad V_{k+1} = \mathrm{Proj}_{\mathcal{V}} T V_k$$

with projection $\mathrm{Proj}_{\mathcal{V}} V = \arg\min_{V' \in \mathcal{V}} \|V - V'\|$ for certain norm $\|\cdot\|$; and

- the approximate policy iteration (API) is written as

$$(\textbf{API}) \quad \pi_{k+1}(s) = \arg\max_a R(s,a) + \gamma \sum_{s'} \mathtt{Pr}(s'|s,a) V(s'),$$

with $V \approx V^{\pi_k}$.

- in particular, the value function V in API is obtained through approximation in a function space $\mathcal{V}$
- expect that if V is close to $V^\star$ then the policy π will be close to optimal

Bounds by Bellman residual

- Apart from bound by approximation error, it is also possible to bound the performance with the Bellman residual as [WB93]

$$\| \underbrace{V^\star - V^\pi}_{\text{performance gap}} \|_\infty \leq \frac{2}{1-\gamma} \| \underbrace{TV - V}_{\text{Bellman residual}} \|_\infty, \tag{13}$$

  where T is the Bellman optimality operator. This follows by combing

$$\begin{aligned}
\|V^\star - V\|_\infty \leq & \|V^\star - TV\|_\infty + \|TV - V\|_\infty \\
\leq & \gamma \|V^\star - V\|_\infty + \|TV - V\|_\infty \leq \frac{1}{1-\gamma} \|TV - V\|_\infty
\end{aligned}$$

  and

$$\begin{aligned}
\|V - V^\pi\|_\infty \leq & \|V - TV\|_\infty + \|TV - V^\pi\|_\infty \\
\leq & \|TV - V\|_\infty + \gamma \|V - V^\pi\|_\infty \quad (\text{By } TV = T^\pi V) \\
\leq & \frac{1}{1-\gamma} \|TV - V\|_\infty
\end{aligned}$$

Bellman residual minimisation

- this Bellman residual bound motivates one to minimise the Bellman residual, which leads to Bellman residual minimisation

$$\textbf{(BRM)} \quad \min_{V \in \mathcal{V}} \| TV - V \|$$

for some norm $\| \cdot \|$

Performance bounds of AVI

- the role of V_k in AVI is similar to that of the **target network** in DQN:
- AVI bound [BT96]: after K iterations, we have

$$\|V^\star - V^{\pi_K}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2} \max_{0\leq k\leq K} \|TV_k - V_{k+1}\|_\infty + \frac{2\gamma^{K+1}}{1-\gamma}\|V^\star - V_0\|_\infty.$$

In particular, if

$$\tilde{V} = \mathrm{Proj}_{\mathcal{V}} T\tilde{V}$$

with $\tilde{\pi}$ being a greedy policy with respect to $R + \gamma P\tilde{V}$, then

$$\|V^\star - V^{\tilde{\pi}}\|_\infty \leq \frac{2}{(1-\gamma)^2} \inf_{V\in\mathcal{V}} \|V^\star - V\|_\infty.$$

Performance bounds of AVI: the proof

- by letting $\varepsilon = \max_{0\le k\le K}\|TV_k - V_{k+1}\|_\infty$, we can derive

$$\begin{aligned}\|V^\star - V_{k+1}\|_\infty \le& \|TV^\star - TV_k\| + \|TV_k - V_{k+1}\|_\infty\\ \le& \gamma\|V^\star - V_k\|_\infty + \varepsilon,\end{aligned}$$

  and thus,

$$\begin{aligned}\|V^\star - V_k\|_\infty \le& (1+\gamma+\cdots+\gamma^{K-1})\varepsilon + \gamma^K\|V^\star - V_0\|_\infty\\ \le& \frac{1}{1-\gamma}\varepsilon + \gamma^K\|V^\star - V_0\|_\infty.\end{aligned}$$

  The first result follows by combining (12)

- let the projection use the infinity norm, then the AVI is contractive with fixed point $\tilde V = \mathrm{Proj}_{\mathcal{V}} T\tilde V$ and we can obtain

$$\|V^\star - V\|_\infty \le \|V^\star - \mathrm{Proj}_{\mathcal{V}} V^\star\|_\infty + \|\mathrm{Proj}_{\mathcal{V}} V^\star - \tilde V\|_\infty$$

  with $\|\mathrm{Proj}_{\mathcal{V}} V^\star - \tilde V\|_\infty = \|\mathrm{Proj}_{\mathcal{V}} TV^\star - \mathrm{Proj}_{\mathcal{V}} T\tilde V\| \le \gamma\|V^\star - \tilde V\|_\infty$. the second result then follows from using (12).

Performance bounds of API

- API bound [BT96]: the asymptotic performance bound is

$$\limsup_{k\to\infty} \|V^\star - V^{\pi_k}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2} \limsup_{k\to\infty} \|V_k - V^{\pi_k}\|_\infty.$$

Performance bounds of API: the proof

- let $e_k = V_k - V^{\pi_k}$ denote the approximation error, $g_k = V^{\pi_{k+1}} - V\pi_k$ the performance gain and $l_k = V^\star - V^{\pi_k}$ the loss of using π_k instead of $\pi^\star$
- we can show that the next policy cannot be much worse than the current one as

$$g_k \geq -\gamma(I - \gamma P^{\pi_{k+1}})^{-1}(P^{\pi_{k+1}} - P^{\pi_k})e_k.$$

- the loss at the next iteration is bounded by the current loss as

$$l_{k+1} \leq \gamma P^{\pi^\star} l_k + f_k$$

where $f_k = \gamma[P^{\pi_{k+1}}(I - \gamma P^{\pi_{k+1}})^{-1}(I - \gamma P^{\pi_k}) - P^{\pi^\star}]e_k$

- by taking the limit on both sides, we can obtain

$$\begin{aligned}(I - \gamma P^{\pi^\star}) \limsup_{k\to\infty} l_k &\leq \limsup_{k\to\infty} f_k \\ \limsup_{k\to\infty} l_k &\leq (I - \gamma P^{\pi^\star})^{-1} \limsup_{k\to\infty} f_k.\end{aligned}$$

This leads to

$$\begin{aligned}\limsup_{k\to\infty} \|l_k\|_\infty &\leq \frac{\gamma}{1-\gamma}\|P^{\pi_{k+1}}(I - \gamma P^{\pi_{k+1}})^{-1}(I - \gamma P^{\pi_k}) - P^{\pi^\star}\|_\infty \|e_k\|_\infty \\ &\leq \frac{\gamma}{1-\gamma}(\frac{1+\gamma}{1-\gamma} + 1)\|e_k\|_\infty + \frac{2\gamma}{(1-\gamma)^2}\|e_k\|_\infty,\end{aligned}$$

which validates the result

Performance bounds of BRM

- BRM [WB93]: if $V_{\mathtt{BRM}} = \arg\min_{V \in \mathcal{V}} \| TV - V \|_\infty$,

$$\| V^\star - V_{\mathtt{BRM}}^{\pi} \| \le \frac{2(1+\gamma)}{1-\gamma} \inf_{V \in \mathcal{V}} \| V^\star - V \|_\infty .$$

Proof. Note that

$$\begin{aligned} \| TV - V \|_\infty \le & \| TV - TV^\star \|_\infty + \| V^\star - V \|_\infty \\ \le & (1+\gamma) \| V^\star - V \|_\infty . \end{aligned}$$

The bound follows by combing

$$\begin{aligned} \| TV_{\mathtt{BRM}}^{\pi} - V_{\mathtt{BRM}}^{\pi} \|_\infty = & \inf_{V \in \mathcal{V}} \| TV - V \|_\infty \\ \le & (1+\gamma) \inf_{V \in \mathcal{V}} \| V^\star - V \|_\infty \end{aligned}$$

and (13).

Sample-based ADP: sample complexity

- RL essentially solve ADP with sampled transition and rewards
- from statistical learning, the *prediction error* of RL comes from two sources:
  1. **approximation error**, error due to the projection operation;
  2. **estimation error**, since both the Bellman operator and projection are evluated with samples.
- error propagation of sample-based ADP can be analyzed by using tools such as McDiarmid's inequality, if

$$\sup_{x_1,\ldots,x_n,x_i'} |f(x_1,\ldots,x_i,\ldots,x_n) - f(x_1,\ldots,x_i',\ldots,x_n)| \leq c_i,$$

  we have

$$\mathtt{Pr}(|f(x_1,\ldots,x_n) - \mathbb{E}[f(x_1,\ldots,x_n)]| \geq \varepsilon) \leq \exp(-\frac{2\varepsilon}{\sum_{i=1}^{n} c_i}).$$

- combing the sample error and performance bounds before, the generalisation bound (performance guarantee) of sample-based ADPs can be derived

Sample complexity: sampling-based AVI

- consider following setup [MS08]: sample i.i.d. n states $s^{(i)} \sim \mu$, and from each state-action pair $s^{(i)}, a$, generate m one-step transition samples from a simulator $s_a^{(i,j)} \sim \mathtt{Pr}(\cdot|s^{(i)}, a)$
- iterate AVI with **fitted value functions** K times:

$$V_{k+1} = \arg\min_{V \in \mathcal{V}} \sum_{i=1}^{n} |V(s^{(i)}) - \max_a [r(s^{(i)}, a) + \frac{\gamma}{m} \sum_{j=1}^{m} V_k(s_a^{(i,j)})]|^2.$$

Sample complexity: sampling-based AVI

- With probability at least $1-\delta$,

$$\|V^\star - V^{\pi_K}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2} C^{1/p} d(T\mathcal{V},\mathcal{V}) + O(\gamma^k) + O\Big(\frac{V(\mathcal{V})\log(1/\delta)}{n}\Big)^{1/4} + O\Big(\frac{\log(1/\delta)}{m}\Big)^{1/2},$$

  where

$$d(T\mathcal{V},\mathcal{V}) := \sup_{V\in\mathcal{V}} \inf_{V'\in\mathcal{V}} \|TV - V'\|_{2,\mu}, \text{ with } \|V\|_{2,\mu} = \Big(\sum_x \mu(x)V(x)^2\Big)^{1/2}$$

  measures the Bellman residual of the space $\mathcal{V}$, the constant C satisfies $1 \leq C \leq \mathtt{Pr}(\cdot|s,a)/\mu(\cdot)$ for any s and a, and $V(\mathcal{V})$ is the capacity measure of $\mathcal{V}$ (i.e., pseudo-dimension[14]).

- In addition to the **AVI** with fitted value iteration [MS08], PAC bounds for finite-time analysis has also been developed for **API** such as LSPTD/LSPI by [LGM12] and BRM-based PI by [Mai+10]

[14] The Pseudo-dimension, also referred to the Pollard dimension, is a generalization of the VC-dimension to real-valued functions.

Deep Q learning and its sample complexity

- finite time bound of Q-learning with non-linear multi-layer ReLU unit and i.i.d. samples have been studied [YXW19]
- the major result states that, with high probability,

$$\|Q^{\pi_K} - Q^\star\|_{1,\mu} \leq C\frac{\phi_{\mu,\sigma}\cdot\gamma}{(1-\gamma)^2}|\mathbb{A}|(\log n)^{1+2\xi^*} n^{(\alpha^*-1)/2} + \frac{4\gamma^{K+1}}{1-\gamma}\cdot\max_{s,a} R(s,a),$$

  where n is the number of samples, $C > 0, \xi^*$ and $\alpha^\star$ is some constant, $\phi_{\mu,\sigma}$ is related to concentration coefficients of underlying Markov chain
- this was further extended to non i.i.d. samples by [XG19]. With high probability, the bound decreases at rate $1/\sqrt{K}$ with sufficiently network width m as

$$\frac{1}{K}\sum_{k=0}^{K}\mathbb{E}[(Q(s,a;\theta_k) - Q^\star(s,a))^2] \leq O\Big(\frac{1}{m^{1/6}} + \frac{1}{\sqrt{K}}\Big)$$

Concluding remarks

References I

Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

Ronald A Howard. *Dynamic Programming and Markov Processes*. MIT Press, 1960.

Alessandro Lazaric, Mohammad Ghavamzadeh, and Rémi Munos. "Finite-sample analysis of least-squares policy iteration". In: *The Journal of Machine Learning Research* 13 (2012), pp. 3041–3074.

Odalric-Ambrym Maillard et al. "Finite-sample analysis of Bellman residual minimization". In: *Asian Conference on Machine Learning (ACML)*. 2010, pp. 299–314.

References II

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

Rémi Munos and Csaba Szepesvári. "Finite-time bounds for fitted value iteration". In: *Journal of Machine Learning Research* 9 (2008), pp. 815–857.

Ŕemi Munos. *Introduction to Reinforcement Learning and multi-armed bandits*. NETADIS Summer School, 2013.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

References III

Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Richard S Sutton et al. “Policy gradient methods for reinforcement learning with function approximation”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 1999, pp. 1057–1063.

Leslie G Valiant. “A theory of the learnable”. In: *Communications of the ACM* 27.11 (1984), pp. 1134–1142.

Christopher John Cornish Hellaby Watkins. “Learning From Delayed Rewards”. PhD Thesis. University of Cambridge, 1989.

References IV

Ronald J. Williams and Leemon C. Baird III. *Tight performance bounds on greedy policies based on imperfect value functions*. Tech. rep. NU-CCS-93-14, College of Computer Science, Northeastern University. 1993.

Ronald J Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Machine learning* 8.3-4 (1992).

Shuang Wu and Jun Wang. *Decision making and AI: a white paper*. 2020.

Pan Xu and Quanquan Gu. “A finite-time analysis of q-learning with neural network function approximation”. In: *arXiv preprint arXiv:1912.04511* (2019).

References V

Zhuoran Yang, Yuchen Xie, and Zhaoran Wang. "A theoretical analysis of deep Q-learning". In: *arXiv preprint arXiv:1901.00137* (2019).