Reinforcement Learning

Master 1 Computer Science

Aurélien Garivier

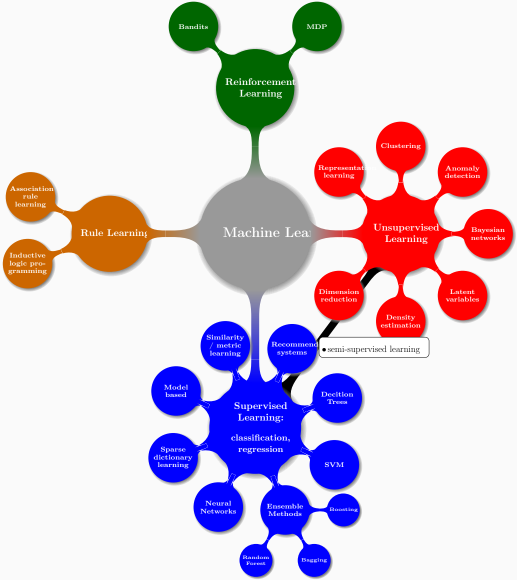
2019-2020

Table of contents

What is Reinforcement Learning?

Outline

Different Types of Learning



Machine Lea

Reinforcement Learning
Bandits
MDP

Unsupervised Learning
Clustering
Representati learning
Anomaly detection
Bayesian networks
Latent variables
Density estimation
Dimension reduction

• semi-supervised learning

Rule Learning
Association rule learning
Inductive logic programming

Supervised Learning:

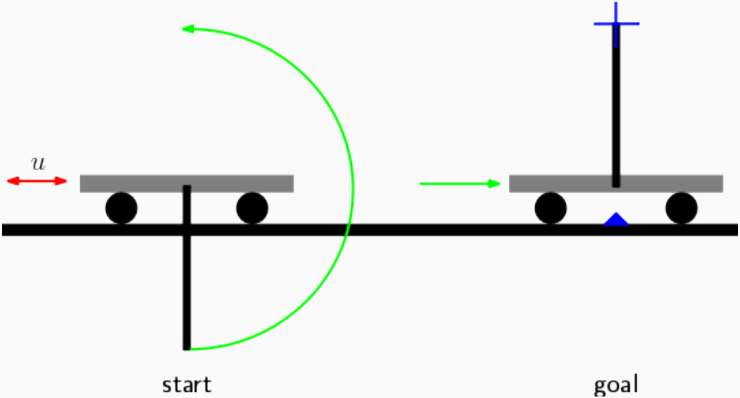
classification, regression
Similarity / metric learning
Recommend systems
Model based
Decition Trees
Sparse dictionary learning
SVM
Neural Networks
Ensemble Methods
Boosting
Random Forest
Bagging

Reinforcement Learning

- Dates back to 1950's (Bellman)
- Stochastic Optimal Control
- Dynamic Programming
- Strong revival with the work of

DeepMind

Example: Inverted Pendulum



The Learning algorithm used by Martin is *Neural Fitted Q iteration*, a version of Q-iteration where neural networks are used as function approximators

Some Applications

- TD-Gammon. [Tesauro '92-'95]: backgammon world champion
- KnightCap [Baxter et al. '98]: chess (2500 ELO)
- Computer poker [Alberta, '08...]
- Computer go [Mogo '06], [AlphaGo '15, Alphazero '18]
- Atari, Starcraft, etc. [Deepmind '10 sqq]
- Robotics: jugglers, acrobots, ... [Schaal et Atkeson '94 sqq]
- Navigation: robot guide in Smithonian Museum [Thrun et al. '99]
- Lift command [Crites et Barto, 1996]
- Internet Packet Routing [Boyan et Littman, 1993]
- Task Scheduling [Zhang et Dietterich, 1995]
- Maintenance [Mahadevan et al., 1997]
- Social Networks [Acemoglu et Ozdaglar, 2010]
- Yield Management, pricing [Gosavi 2010]
- Load forecasting [S. Meynn, 2010]
- ...

Outline

What is Reinforcement Learning?

- Introduction
- Reinforcement Learning Framework

Policy Evaluation

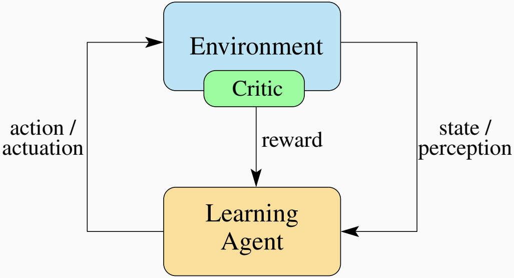
- Bellman's Equation for a Policy
- Optimal Policies

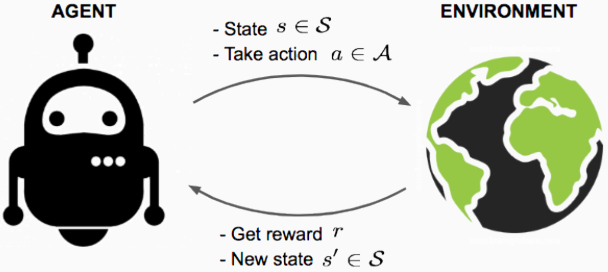
Planning

Learning

- The Q-table
- Model-free Learning: Q-learning

A Model for RL: MDP





exploration
vs
exploitation
dilemma

src: `https://lilianweng.github.io`

Model: Markov Decision Process

Markov Decision Process = 4-uple $(\mathcal{S}, \mathcal{A}, k, r)$:

- State space $\mathcal{S} = \{1, \dots, p\}$
- Action space $\mathcal{A} = \{1, \dots, K\}$
- Transition kernel $k \in \mathfrak{M}_1(\mathcal{S})^{\mathcal{S}\times\mathcal{A}}$
- Random reward function $r \in \mathfrak{M}_1(\mathbb{R})^{\mathcal{S}\times\mathcal{A}}$

Dynamic = controlled Markov Process:

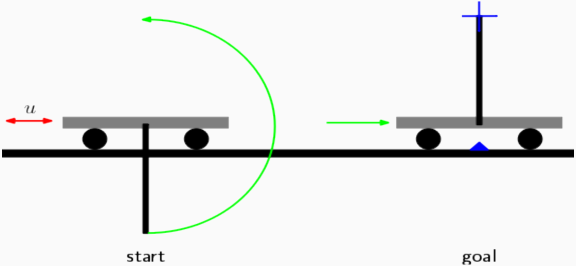
- Initial state S_0
- At each time $t \in \mathbb{N}$:
  - choose action A_t
  - get reward $X_t \sim r(\cdot|S_t, A_t)$
  - switch to new state $S_{t+1} \sim k(\cdot|S_t, A_t)$

Cumulated reward: $W = \sum_{t=0}^{\infty} \gamma^t X_t$ where $\gamma \in (0, 1)$ is a *discount parameter*

Goal

choose the actions so as to **maximize the cumulated reward** in expectation.

Example: inverted pendulum



- <u>State</u>: horizontal position, angular position and velocity
  State space: $\mathcal{S} = [0,1] \times [-\pi,\pi] \times \mathbb{R}$
- <u>Action</u>: move left or right
  Action space: $\mathcal{A} = \{-1,+1\}$
- <u>Reward</u> = proportional to height of the stick end: if $S_t = (x_t, \theta_t, \dot{\theta}_t)$,

$$X_t = \sin(\theta_t)$$

- <u>Transition</u>: given by the laws of physics

Example: Retail Store Management 1/2

You owe a bike store. During week t, the (random) demand is D_t units. On Monday morning you may choose to command A_t additional units: they are delivered immediately before the shop opens. For each week:

- Maintenance cost: $h(s)$ for s units in stock left from the previous week
- Command cost: $C(a)$ for a units
- Sales profit: $f(q)$ for q units sold
- Constraint:
  - your warehouse has a maximal capacity of M unit (any additionnal bike gets stolen)
  - you cannot sell bikes that you don't have in stock

Example: Retail Store Management 2/2

- State: number of bikes in stock on Sunday
  State space: $\mathcal{S} = \{0, \dots, M\}$
- Action: number of bikes commanded at the beginning of the week
  Action space: $\mathcal{A} = \{0, \dots, M\}$
- Reward = balance of the week: if you command A_t bikes,

$$X_t = -C(A_t) - h(S_t) + f\big(\min(D_t, S_t + A_t, M)\big)$$

- Transition: you end the week with

$$S_{t+1} = \max\big(0, \min(M, S_t + A_t) - D_t\big) \qquad \text{bikes}$$

We may assume for example that $h(s) = h \cdot s$, $f(q) = p \cdot q$ and $C(a) = c_0 \mathbb{1}\{a > 0\} + c \cdot a$

Policies: Controlled Markov Chain

Policy $\pi : \mathcal{S} \to \mathcal{A}$
$\pi(s)$ = action chosen every time the agent is in state s

- can be randomized $\pi : \mathcal{S} \to \mathfrak{M}_1(\mathcal{A})$

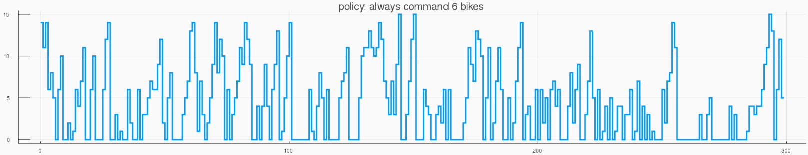
  $\pi(s)_a$ = probability to choose action a in state s
- can be non-stationary $\pi : \mathcal{S} \times \mathbb{N} \to \mathfrak{M}_1(\mathcal{A})$

  $\pi(s, t)_a$ = probability to choose action a in state s at time t
- . . . but it is useless: stationary, deterministic policies can do as well

For a given policy π, the sequence of states $(S_t)_{t\geq 0}$ is a Markov chain of kernel K_π:

$$K_\pi(s, s') = k\big(s'|s, \pi(s)\big)$$

and the sequence of rewards $(X_t)_{t\geq 0}$ is a hidden Markov chain



Policy Evaluation

Outline

- What is Reinforcement Learning?
  - Introduction
  - Reinforcement Learning Framework
- Policy Evaluation
  - Bellman's Equation for a Policy
  - Optimal Policies
- Planning
- Learning
  - The Q-table
  - Model-free Learning: Q-learning

Policy Value Function

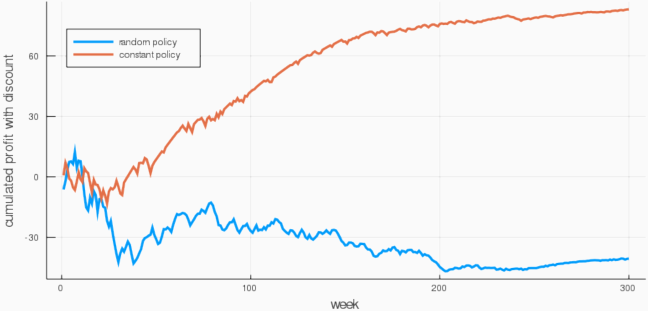
Avg reward function $\bar{r}(s,a) = \mathbb{E}\big[X_t \big| S_t = s, A_t = a\big] = \text{mean of } r(\cdot|s,a)$

The **value function** of π is $V_\pi : \mathcal{S} \to \mathbb{R}$ defined by

$$V_\pi(s) = \mathbb{E}_\pi\left[\sum_{t\geq 0} \gamma^t X_t \middle| S_0 = s\right]$$

$$= \bar{r}\big(s,\pi(s)\big) + \gamma \sum_{s_1} k\big(s_1|s,\pi(s)\big)\,\bar{r}\big(s_1,\pi(s_1)\big) + \gamma^2 \sum_{s1,s2} k\big(s_1|s,\pi(s)\big)\,k\big(s_2|s_1,\pi(s_1)\big)\,\bar{r}\big(s_2,\pi(s_2)\big) + \dots$$

One can simulate runs of the policy and estimate V_π by Monte-Carlo



Bellman's Equation for a Policy

Average reward function for policy π: $\bar{R}_\pi = \left[s \mapsto \bar{r}(s, \pi(s))\right]$

Matrix notation: identify functions $\mathcal{S} \to \mathbb{R}$ with $\mathbb{R}$-valued vectors
Coordinatewise partial order: $\forall U, V \in \mathbb{R}^{\mathcal{S}}, U \leq V \iff \forall s \in \mathcal{S}, U_s \leq V_s$

Bellman's Equation for a policy

The values $V_\pi(s)$ of a policy π at states $s \in \mathcal{S}$ satisfy the linear system:

$$\forall s \in \mathcal{S}, V_\pi(s) = \bar{r}\big(s, \pi(s)\big) + \gamma \sum_{s' \in \mathcal{S}} k\big(s' | s, \pi(s)\big)\, V_\pi(s')$$

In matrix form:

$$V_\pi = \bar{R}_\pi + \gamma K_\pi V_\pi$$

Theorem

Bellman's equation for a policy admits a unique solution given by

$$V_\pi = (I_{\mathcal{S}} - \gamma K_\pi)^{-1} \bar{R}_\pi$$

Operator View

Bellman's Transition Operator

Bellman's Transition Operator $T_\pi : \mathbb{R}^{\mathcal{S}} \to \mathbb{R}^{\mathcal{S}}$ is defined by

$$T_\pi(V) = \bar{R}_\pi + \gamma K_\pi V$$

It is **affine**, **isotonic** ($U \leq V \implies T_\pi U \leq T_\pi V$) and γ-**contractant**: $\forall U, V \in \mathbb{R}^{\mathcal{S}}, \|T_\pi U - T_\pi V\|_\infty \leq \gamma \|U - V\|_\infty$

Proof in exercise

Thus, T_π **has a unique fixed point equal to** V_π
Moreover, for all $V_0 \in \mathbb{R}^{\mathcal{S}}$, $T_\pi^n V_0 \underset{n\to\infty}{\to} V_\pi$.
Proof in exercise
Also note that

$$\begin{aligned} T_\pi^n V_0 &= \bar{R}_\pi + \gamma K_\pi R_\pi + \dots + \gamma^n K_\pi^n R_\pi + \gamma^n K_\pi^n V_0 \\ &\to \left(I_{\mathcal{S}} + \gamma K_\pi + \gamma^2 K_\pi^2 + \dots\right) \bar{R}_\pi = (I_{\mathcal{S}} - \gamma K_\pi)^{-1} \bar{R}_\pi = V_\pi \end{aligned}$$

Sample-based Policy Evaluation: $TD(0)$

As an alternative to plain Monte-Carlo evaluation, the **Temporal Difference** method is based on the idea of *stochastic approximation*

Algorithm 1: $TD(0)$

```
Input : $V_0$ = any function (e.g. $V_0 \leftarrow 0_{\mathcal{S}}$)
        $T$ = number of iterations
$V \leftarrow V_0$
for $t \leftarrow 0$ to $T$ do
    $r' \leftarrow \mathrm{reward}(s, \pi(s))$
    $s' \leftarrow \mathrm{next\_state}(s, \pi(s))$
    $V(s) \leftarrow (1 - \alpha_t)V(s) + \alpha_t\big(r' + \gamma V(s')\big)$
end
Return: $V$
```

Stochastic Approximation

Let $(X_n)_{n\geq 1}$ be a sequence of iid variables with expectation μ. A *sequential estimator* of μ is: $\hat{\mu}_1 = X_1$ and for all $n \geq 2$,

$$\hat{\mu}_n = (1-\alpha_n)\hat{\mu}_{n-1} + \alpha_n X_n$$

Proposition

When $(\alpha_n)_n$ is a decreasing sequence such that $\sum_n \alpha_n = \infty$ and $\sum_n \alpha_n^2 < \infty$, if the $(X_n)_n$ have a finite variance, $\hat{\mu}_n$ converges almost-surely to μ.

Case $\alpha_n = \frac{1}{n}$: $\hat{\mu}_n = \frac{X_1 + \cdots + X_n}{n}$ and $\mathbb{E}\left[(\hat{\mu}_n - \mu)^2\right] = \frac{\mathbb{V}ar[X_1]}{n}$

In $TD(0)$: $V(s) \leftarrow (1-\alpha_t)V(s) + \alpha_t\big(r' + \gamma V(s')\big)$

At every step, if $V = V_\pi$ then the expectation of the rhs is equal to $V(s)$

Outline

- What is Reinforcement Learning?
    - Introduction
    - Reinforcement Learning Framework
- Policy Evaluation
    - Bellman's Equation for a Policy
    - Optimal Policies
- Planning
- Learning
    - The Q-table
    - Model-free Learning: Q-learning

What are Optimal Policies – and How to Find them?

Goal

Among all possible policies $\pi : \mathcal{S} \to \mathcal{A}$, find an *optimal* one π^* maximizing the expected value *on all states at the same time*:

$$\forall \pi : \mathcal{S} \to \mathcal{A}, \forall s \in \mathcal{S} : V_{\pi^*}(s) \geq V_\pi(s)$$

Questions:

- Is there always an optimal policy π^*?
- How to find π^*...
  - ...when the model (k, r) is known?
    → *planning*
  - ...when the model is unknown, but only sample trajectories can be observed?
    → *learning*

Bellman's Optimality Operator

Bellman's Optimality Operator

Bellman's Optimality Operator $T_* : \mathbb{R}^{\mathcal{S}} \to \mathbb{R}^{\mathcal{S}}$ defined by

$$\big(T_*(V)\big)_s = \max_{a \in \mathcal{A}} \left\{ \bar{r}(s,a) + \gamma \sum_{s' \in \mathcal{S}} k(s'|s,a) V_{s'} \right\}$$

is **isotonic** and γ-**contractant**. Besides, for every policy π, $T_\pi \leq T_*$ in the sense that $\forall U \in \mathbb{R}^S$, $T_\pi U \leq T_* U$

Note that T_* is not affine, due to the presence of the max

Proof in exercise

Policy Improvement

Greedy Policy

For every $V \in \mathbb{R}^{\mathcal{S}}$, there exist at least one policy π such that $T_\pi V = T_* V$. It is called **greedy w.r.t.** V, and is characterized as:

- $\forall s \in \mathcal{S}, \pi(s) \in \arg\max_{a \in \mathcal{A}} \left\{ \bar{r}(s,a) + \gamma \sum_{s' \in \mathcal{S}} k(s'|s,a) V_{s'} \right\}$
- $\pi \in \arg\max_{\pi'} \bar{R}_\pi + \gamma K_\pi V$

Policy Improvement Lemma

For any policy π, any greedy policy π' wrt V_π improves on π: $V_{\pi'} \geq V_\pi$

Proof in exercise

Optimal Value Function

Since T_* is γ-contractant, it **has a unique fixed point** V_* and $\forall V \in \mathbb{R}^{\mathcal{S}}, T_*^n V \underset{n \to \infty}{\to} V_*$

Bellman's Optimality Theorem

V_* is the optimal value function:

$$\forall s \in \mathcal{S}, V_*(s) = \max_\pi V_\pi(s)$$

and any policy π such that $T_\pi V_* = V_*$ is optimal

Proof in exercise

Corollary

Any finite MDP admits an optimal (deterministic and stationary) policy

This optimal policy is not necessarily unique

Planning

Value Iteration

If you know V_*, computing the greedy policy w.r.t V_* gives an optimal policy. And V_* is the fixed point of Bellman's optimality operator T_*, hence can be computed by a simple iteration process:

```
Algorithm 2: Value Iteration
Input  : $\epsilon$ = required precision, $V_0$ = any function (e.g. $V_0 \leftarrow 0_{\mathcal{S}}$)
1 $V \leftarrow V_0$
2 while $\|V - T_*(V)\| \geq \frac{(1-\gamma)\epsilon}{\gamma}$ do
3     $V \leftarrow T_* V$
4 end
Return: $T_* V$
```

Theorem

The Value Iteration algorithm returns a value vector V such that $\|V - V_*\|_\infty \leq \epsilon$ using at most $\frac{\log \frac{M}{(1-\gamma)\epsilon}}{1-\gamma}$ iterations where $M = \|T_* V_0 - V_0\|_\infty$

Remark: if V_0 is the value function of some policy π_0 and if π_t is the sequence of policies obtained on line 3 (i.e. π_t is the greedy policy w.r.t. V_{t-1}), then the returned function obtained after T iterations is the value of the (non-stationary) policy $(\pi'_t)_t$, where $\pi'_t = \pi_{(T-t)_+}$.

Proof

Proof in exercise

Policy Iteration

The Policy Improvement lemma directly suggests Policy Iteration: starting from any policy, evaluate it (by solving the linear system $T_\pi V_\pi = V_\pi$) and improve π greedily:

Algorithm 3: Policy Iteration

```
Input : $\pi_0$ = any policy (e.g. chosen at random)
$\pi \leftarrow \pi_0$
$\pi' \leftarrow \mathrm{NULL}$
while $\pi \neq \pi'$ do
    compute $V_\pi$
    $\pi' \leftarrow \pi$
    $\pi \leftarrow$ greedy policy w.r.t. $V_\pi$
end
Return: $\pi$
```

NB: the iterations of PI are much more costly than those of VI

Convergence of Policy Iteration

Theorem

The Policy Iteration algorithm always returns an optimal policy in at most $|\mathcal{A}|^{|\mathcal{S}|}$ iterations.

Proof: the Policy Improvement lemma shows that the value of π raises strictly at each iteration before convergence, and there are only $|\mathcal{A}|^{|\mathcal{S}|}$ different policies. Remark: better upper-bounds in $O\left(\frac{|\mathcal{A}|^{|\mathcal{S}|}}{|\mathcal{S}|}\right)$ are known.

Lemma

Let (U_n) be the sequence of value functions generated by the Value Iteration algorithm, and (V_n) be the one for the Policy Iteration algorithm. If $U_0 = V_0$ (i.e. if U_0 is the value function of π_0), then

$$\forall n \geq 0, U_n \leq V_n$$

Proof: Assume by induction that $U_n \leq V_n$. Since T_* and $T_{\pi_{n+1}}$ are isotonic, and since $V_n \leq V_{n+1}$ by the policy improvement lemma:

$$U_{n+1} = T_* U_n \leq T_* V_n = T_{\pi_{n+1}} V_n \leq T_{\pi_{n+1}} V_{n+1} = V_{n+1}$$

Linear Programming

Proposition

Let $\alpha : \mathcal{S} \to (0, +\infty)$. V_* is the only solution of the linear program

$$\min_V \sum_{s \in \mathcal{S}} \alpha(s) V(s)$$

$$\text{subject to } \forall s \in \mathcal{S}, \forall a \in \mathcal{A}, V(s) \geq \bar{r}(s,a) + \gamma \sum_{s' \in \mathcal{S}} k(s'|s,a) V(s')$$

Proof: By Bellman's optimality equation $T_* V_* = V_*$, V_* satisfies the constraint with equality. If V satisfies the condition, then $W = V - V_*$ is such that $\forall s, a, W(s) \geq \gamma \sum_{s' \in \mathcal{S}} k(s'|s,a) W(s')$; thus if $s_- \in \arg\min_{s \in \mathcal{S}} W(s)$ one gets $W(s_-) \geq \gamma \sum_{s' \in \mathcal{S}} k(s'|s,a) W(s') \geq -\gamma \big|W(s_-)\big|$, hence $W(s_-) \geq 0$ and $W \geq 0$, and thus $\sum_{s \in \mathcal{S}} \alpha(s) V(s) \geq \sum_{s \in \mathcal{S}} \alpha(s) V_*(s)$ with equality iff $V = V_*$.

This linear program has $|\mathcal{S}| \cdot |\mathcal{A}|$ rows (constraints) and $|\mathcal{S}|$ columns (variables). Solvers have a complexity typically larger in the number of rows than columns. Hence, it may be more efficient to consider the dual problem.

Learning

Outline

What is Reinforcement Learning?

- Introduction
- Reinforcement Learning Framework

Policy Evaluation

- Bellman's Equation for a Policy
- Optimal Policies

Planning

Learning

- The Q-table
- Model-free Learning: Q-learning

State-Action Value Function

Definition

The state-action value function $Q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ for policy π is the expected return for first taking action a in state s, and then following policy π:

$$\begin{aligned} Q_\pi(s,a) &= \text{``}\mathbb{E}_{a,\pi}\text{``} \left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \,\middle|\, S_0 = s, A_0 = a \right] \\ &= \bar{r}(s,a) + \gamma \sum_{s'} k(s'|s,a) V_\pi(s') \end{aligned}$$

The state-action value function is a key-tool in the study of MDPs

Observe that $Q_\pi\big(s, \pi(s)\big) = V_\pi(s)$.

Policy Improvement Lemma

Lemma

For any two policies π and π',

$$\Big[\forall s \in \mathcal{S}, Q_\pi\big(s, \pi'(s)\big) \geq Q_\pi\big(s, \pi(s)\big)\Big] \implies \Big[\forall s \in \mathcal{S}, V_{\pi'}(s) \geq V_\pi(s)\Big]$$

Furthermore, if one of the inequalities in the LHS is strict, then at least one of the inequalities in the RHS is strict

Proof in exercise

Bellman's Optimality Condition: Q-table formulation

Theorem

A policy π is optimal if and only if

$$\forall s \in \mathcal{S},\ \pi(s) \in \arg\max_{a \in \mathcal{A}} Q_\pi(s, a)$$

Proof in exercise

Outline

What is Reinforcement Learning?

- Introduction
- Reinforcement Learning Framework

Policy Evaluation

- Bellman's Equation for a Policy
- Optimal Policies

Planning

Learning

- The Q-table
- Model-free Learning: Q-learning

Q-Learning

```
Algorithm 4: $Q$-learning
Input : $Q_0 =$ any state-value function (e.g. chosen at random)
        $s_0 =$ initial state (possibly chosen at random)
        $\pi =$ learning policy (may be $\epsilon$-greedy w.r.t. current $Q$)
        $T =$ number of iterations
1 $Q \leftarrow Q_0$
2 $s \leftarrow s_0$
3 for $t \leftarrow 0$ to $T$ do
4     $a \leftarrow \mathrm{select\_action}(\pi(Q), s)$
5     $r' \leftarrow \mathrm{random\_reward}(s, a)$
6     $s' \leftarrow \mathrm{next\_state}(\mathrm{s}, \mathrm{a})$
7     $Q(s,a) \leftarrow Q(s,a) + \alpha_t \left[ r' + \gamma \max_{a' \in \mathcal{A}} Q(s', a') - Q(s,a) \right]$
8     $s \leftarrow s'$
9 end
Return: $Q$
```

Off-policy learning: update rule $\neq$ learning policy (on l.7, a' may be different from played action a)

Convergence of Q-learning

Denote by $(S_t)_t$ (resp. $(A_t)_t$) the sequence of states (resp. actions) visited by the Q-learning algorithm. For all $(s, a) \in \mathcal{S} \times \mathcal{A}$, let $\alpha_t(s, a) = \alpha_t \mathbb{1}\{S_t = s, A_t = a\}$

Theorem

If for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$ it holds that $\sum_{t \geq 0} \alpha_t(s, a) = +\infty$ and $\sum_{t \geq 0} \alpha_t^2(s, a) < +\infty$, then with probability 1 the Q-learning algorithm converges to the optimal state-value function Q_*

This condition implies in particular that the policy *select_action* guarantees an infinite number of visits to all state-action pairs (s, a)

The **proof** is more involved, and based on the idea of **stochastic approximation**

SARSA

```
Algorithm 5: SARSA
Input : $Q_0$ = any state-value function (e.g. chosen at random)
        $s_0$ = initial state (possibly chosen at random)
        $\pi$ = learning policy (may be $\epsilon$-greedy w.r.t. current $Q$)
        $T$ = number of iterations
$Q \leftarrow Q_0$
$s \leftarrow s_0$
$a \leftarrow \mathrm{select\_action}(\pi(Q), s)$
for $t \leftarrow 0$ to $T$ do
    $r' \leftarrow \mathrm{random\_reward}(s, a)$
    $s' \leftarrow \mathrm{next\_state}(s, a)$
    $a' \leftarrow \mathrm{select\_action}(\pi(Q), s')$
    $Q(s,a) \leftarrow Q(s,a) + \alpha_t \left[ r' + \gamma Q(s',a') - Q(s,a) \right]$
    $s \leftarrow s'$ and $a \leftarrow a'$
end
Return: $Q$
```

On-policy learning: update rule = learning policy

Q-learning with function approximation

If $\mathcal{S} \times \mathcal{A}$ is large, it is necessary

- to do **state aggregation**
- or to assume a model $Q_\theta(s,a)$ for $Q(s,a)$, where θ is a (finite-dimensional) parameter to be fitted. The obvious extension of Q-learning is:

$$\theta_{t+1} = \theta_t + \alpha_t \big[r' + \gamma \max_{a' \in \mathcal{A}} Q(s',a') - Q(s,a) \big] \, \nabla_\theta Q_{\theta_t}(S_t, A_t)$$

For example, with a linear approximation method with $Q_\theta = \theta^T \phi$ with features map $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$, line 8 of Q-learning is replaced by:

$$\theta \leftarrow \theta + \alpha \big[r' + \gamma \max_{a' \in \mathcal{A}} \theta^T \phi(s',a') - \theta^T \phi(s,a) \big] \, \phi(s,a)$$

- possibility to use any function approximator, typically *splines* or *neural networks*
- ...but very unstable and few guarantees of convergence!
- possiblity to update θ in *batch* and not at each step

Conclusion: What more?

- a lot !
- $TD(\lambda)$ and eligibility traces
- Model-based learning: KL-UCRL

  Build optimistic estimates of Q-table, and play greedily w.r.t. these estimates
- POMDP: Partially Observed Markov Decision Process
- Bandit models

  = MDPs with only 1 state, but already a dilemma exploration vs exploitation
- MCTS: AlphaGo / AlphaZero

References

- **C. Szepesvári** Algorithms for Reinforcement Learning. Morgan & Claypool, 2010
- **M. Mohri, A. Rostamizadeh and A. Talwalkar** *Foundations of Machine Learning, 2nd Ed.*, MIT Press, 2018
- **T. Lattimore and C. Szepesvári** *Bandit Algorithms*, Cambridge Univeristy Press, 2019
- **D. Bertsekas and J. Tsitsiklis** *Neuro-Dynamic Programming*, Athena Scientific, 1996
- **M. L. Puterman.** *Markov Decision Processes, Discrete Stochastic Dynamic Programming.* Wiley-Interscience, 1994.
- **R. S. Sutton & A. G. Barto.** *Reinforcement Learning, an Introduction (2nd Ed.)* MIT Press, 2018.