Evaluation
Online learning
links with optimization and games
Université Paris–Saclay

adaptive diagonal scalings for q-learning

This project requires familiarity with reinforcement learning[1]

Let $\lambda \in (0,1)$. Consider a Markov decision process $(\mathscr{S}, \mathscr{A}, \mathscr{R}, p)$ where $\mathscr{S}$ is the set of states, $\mathscr{A}$ the set of actions, $\mathscr{R} \subset \mathbb{R}$ the set of possible rewards and $p : \mathscr{S} \times \mathscr{A} \times \mathscr{S} \times \mathscr{R}$ the transition function where

$$p(r, s'|s, a) := p(s, a, r, s'), \qquad (s, a, r, s') \in \mathscr{S} \times \mathscr{A} \times \mathscr{R} \times \mathscr{S},$$

corresponds to the probability of obtaining reward r and moving to state s' when action a is chosen at state s. All sets are finite.

An action-value function is a vector $q = (q(s,a))_{(s,a)\in\mathscr{S}\times\mathscr{A}} \in \mathbb{R}^{\mathscr{S}\times\mathscr{A}}$. The Bellman optimality operator (for action-value functions) $\mathrm{B}_* : \mathbb{R}^{\mathscr{S}\times\mathscr{A}} \to \mathbb{R}^{\mathscr{S}\times\mathscr{A}}$ is defined as

$$(\mathrm{B}_* q)(s,a) = \sum_{(r,s')\in\mathscr{S}\times\mathbb{R}} p(r, s'|s, a)\left(r + \lambda \max_{a'\in\mathscr{A}} q(s', a')\right), \qquad (s,a) \in \mathscr{S} \times \mathscr{A},$$

[1] see e.g. https://joon-kwon.github.io/rl-ups/reinforcement-learning-lecture-notes.pdf

where we simply denote $B_* q$ instead of $B_*(q)$. B_* is known to be a contraction: it thus admits a unique fixed point q_*, which is the optimal action-value function.

Without knowledge of p, evaluations of the map B_* cannot be computed, but a stochastic estimator can be obtained as follows. For $q \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ and $(s, a) \in \mathcal{S} \times \mathcal{A}$, if $(R, S') \sim p(\,\cdot\,|s, a)$, in other words if R, S' are the actual (random) reward and new state obtained by choosing action a at state s, then

$$(\hat{B}_* q)(R, S') = R + \lambda \max_{a\in\mathcal{A}} q(S', a)$$

is an unbiaised estimator of $(B_* q)(s, a)$.

Traditional Q-learning is defined as follows. Let $q_0 \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ be an initial action-value function. For all $t \geqslant 0$, let (S_t, A_t, R_t, S'_t) be such that $(R_t, S'_t)|S_t, A_t \sim p(\,\cdot\,|S_t, A_t)$ (often, $S'_t = S_{t+1}$, unless the episode terminates), and

$$q_{t+1}(s, a) = \begin{cases} (1-\gamma_t) q_t(s, a) + \gamma_t \left((\hat{B}_* q_t)(R_t, S'_t)\right) & \text{if } (s, a) = (S_t, A_t) \\ q_t(s, a) & \text{otherwise,} \end{cases}$$

where $\gamma_t \in (0, 1)$. Q-learning is therefore an asynchronous[2] stochastic fixed point iteration.

1) Similarly to the way AdaGrad-Norm is used to solve fixed point problems, define AdaGrad-Diagonal in the context of Q-learning.

Numerous variants of AdaGrad have achieved great success in deep learning optimization. We here consider RMSprop and Adam. Let $d \geqslant 1$ and $x_0 \in \mathbb{R}^d$. For a sequence $(u_t)_{t\geqslant 0}$ in $\mathbb{R}^d$, $\gamma > 0$, the associated RMSprop (resp. Adam) iterates are defined component-wise for $t \geqslant 0$, and $0 \leqslant i \leqslant d$ as

$$x_{t+1,i} = x_{t,i} + \frac{\gamma}{\sqrt{\sum_{\tau=0}^{t} \beta^{t-\tau} u_{\tau,i}^2}} u_{t,i},$$

$$\left(\text{resp.} \quad x_{t+1,i} = x_{t,i} + \frac{\gamma}{\sqrt{\sum_{\tau=0}^{t} \beta_2^{t-\tau} u_{\tau,i}^2}} \sum_{\tau=0}^{t} \beta_1^{t-\tau} u_{\tau,i}\right),$$

where $\beta = .99$, $\beta_1 = .9$ and $\beta_2 = .999$ are common default values.

2) Adapt RMSprop and Adam to the context of Q-learning.

[2] meaning that not all components are updated at each iteration

3) Perform numerical experiments to compare the performance of the above algorithms with traditionnal Q-learning. Consider environments with finite number of states and actions from e.g. the Gymnasium package[3].

[3] https://gymnasium.farama.org/