MS&E338 Reinforcement Learning Lecture 5 - 04/16/2018

Real-Time Value Iteration, Episodic Q-learning and SARSA

Lecturer: Ben Van Roy *Scribe: Danqi Luo, Jordi Feliu Faba*

1 Q-learning update

$$\begin{aligned} Q(s,a) :=& (1-\gamma)Q(s,a) + \gamma(r + \max_{a' \in A} Q(s',a')) \\ =& (1-\gamma)Q(s,a) + \gamma((FQ)(s,a) + \text{zero-mean noise}) \end{aligned}$$

Take the following update rule

$$Q := (1-\gamma)Q + \gamma(FQ + \text{zero-mean noise})$$

Last time we showed that the following proposition is true

Proposition 1. *If* $\forall Q, \bar{Q}, \ \left\|FQ - F\bar{Q}\right\|_2 \leq \alpha \left\|Q - \bar{Q}\right\|_2$ *for some* $\alpha \in (0,1)$*, and we apply stochastic approximation to value iteration with step sizes* $\{\gamma_k\}$ *satisfying* $\sum \gamma_k = \infty$ *and* $\sum {\gamma_k}^2 < \infty$*, then* $Q \to Q^*$

This works also for wighted maximum norm contractions instead of euclidian norm contractions. Even for asynchronous updates, we can get convergence with appropriate conditions.

In order to analyze convergence for asynchronous iterations we must define two types of asynchronism:

- **Total asynchronism:** update each state-action pair i.o.
- **Partial asynchronism:** models of asynchronous iteration that are not adverserial. For instance, models that define upper bounds on the time between updates of $Q(s,a)$ $\forall (s,a)$ or models that guides when you can update a particular $Q(s,a)$ based on a stochastic process.

In HW2 we showed that with total asynchronism, asynchronous iteration of an euclidian norm contraction may fail to converge. With partial asynchronism, this problem can still hold, however under some circumstances we can remove it.

Take the update rule

$$Q := (1-\gamma_k)Q + \gamma_k FQ \text{ with } \sum \gamma_k = \infty, \sum {\gamma_k}^2 < \infty$$

Under partial asynchronism the update rule above will always converge ($Q \to Q^*$) for euclidian norm contractions.

In class a question was raised about how to choose γ_k for a particular problem in the update rule

$$Q := Q + \gamma_k(FQ - Q + \text{noise})$$

Consider $x_1, x_2, x_3, ...$ an i.i.d. sequence. The sample average is the mean x_k. Consider then

$$\bar{x}_{k+1} = \frac{k}{k+1}\bar{x}_k + \frac{1}{k+1}x_{k+1} = \bar{x}_k + \frac{1}{k+1}(x_{k+1} - \bar{x}_k)$$

In this case $\gamma_k = \frac{1}{k+1}$. Similarly for the Q update one could set $\gamma_k = \frac{1}{k+1}$. However it would be too conservative if FQ is far from Q^* and convergence would be too slow. A better option is to use $\gamma_k = \frac{M}{k+1+M}$ for some M. If we never want to stop moving we can set $\gamma_k = \max(\frac{M}{k+1+M}, \epsilon)$.

2 Real-time Value Iteration

Algorithm 1 Real-time Value Iteration

Initialize with some $Q_{0,0}$
for ℓ=0,1,2,$\cdots$ **do**
  sample $s_0 \sim \rho$
  for t=0,1,2,$\cdots$,$\tau-1$ **do**
    select $a_t \in \arg\max_{a\in A} Q_{\ell,t}(s_t,a)$
    sample $s_{t+1} \sim P_{s_t,a_t}$
    $Q_{\ell,t+1}(s,a) = \begin{cases} FQ_{\ell,t}(s,a) & \text{if } (s,a)=(s_t,a_t) \\ Q_{\ell,t}(s,a) & \text{otherwise} \end{cases}$
  end for
  $Q_{\ell+1,0} = Q_{\ell,\tau}$
end for

Proposition 2. *If $Q_{0,0} \geq Q^*$, then*
(1) $\forall s \in S$ visited i.o. $V^{\pi_{\ell,t}} \to V^*(s)$
(2) $\forall s,a$ visited i.o. $Q_{\ell,t}(s,a) \to Q^*(s,a)$

In Real-time Value Iteration applying F requires the knowledge of the MDP (use the knowledge of reward distribution and transition probabilities), which makes it not really a reinforcement learning algorithm. Next, we will present reinforcement learning algorithms where we observe data instead of knowing the MDP. This is so-called Episodic Q-learning, known as one of the "**off-policy**" algorithms. Later we will introduce SARSA, which is known as one of the "**on-policy**" algorithms.

3 Episodic Q-learning

Algorithm 2 Episodic Q-learning

Initialize with some $Q_{0,0}$
for ℓ=0,1,2,$\cdots$ **do**
  sample $s_0 \sim \rho$
  for t=0,1,2,$\cdots$,$\tau-1$ **do**
    select $a_t \in \arg\max_{a\in\mathcal{A}} Q_{\ell,t}(s_t,a)$
    observe $s_{t+1} \sim \mathcal{P}_{s_t,a_t}$
    observe $r_{t+1} \sim \mathcal{R}_{s_t,a_t}$
    $Q_{\ell,t+1}(s,a) = \begin{cases} (1-\gamma_\ell)Q_{\ell,t}(s,a) + \gamma_\ell(r_{t+1} + \max_{a'\in\mathcal{A}} Q_{\ell,t}(s_{t+1},a')), & \text{if } (s,a)=(s_t,a_t) \\ Q_{\ell,t}(s,a), & \text{otherwise} \end{cases}$
  end for
  $Q_{\ell+1,0} = Q_{\ell,\tau}$
end for

In the Episodic Q-learning, we don't use the knowledge of reward distribution and transition probabilities. We do update based on one particular realization (by simulation) of the state-action pair.

Episodic Q-learning doesn't guarantee that you behave optimally. Imagine that for some state $\bar{s}$ we take action $\bar{a}$ and we get a very negative reward. We may never visit the state-action pair $(\bar{s},\bar{a})$ again because of this negative reward. This happens because of using arg max to select the action. This issue would lead to the topic of exploration vs. exploitation in reinforcement learning.

4 Topic of Exploration

If we don't explore at all, then we choose $a_t \in \arg\max_{a\in\mathcal{A}} Q(s_t, a)$. If we do explore, a commonly used exploration strategy is dithering. Here we consider two dithering schemes: *ϵ-greedy* and *Boltzmann* exploration.

1. ϵ-greedy exploration:

$$a_t = \begin{cases} \arg\max_{a\in\mathcal{A}} Q(s_t, a), & w.p.\ 1-\epsilon \\ \text{unif}(\mathcal{A}), & w.p.\ \epsilon\,. \end{cases}$$

2. Boltzmann exploration:

$$a_t \sim \frac{e^{Q(s_t,\cdot)/\beta}}{\sum_{a\in\mathcal{A}} e^{Q(s_t,a)/\beta}}.$$

   Note that here the probability of choosing an action a is exponentially increasing with the Q-value.

One proposition following these two algorithms:

Proposition 3. *In the framework of Episodic Q-learning with ϵ-greedy or Boltzmann exploration, if $\epsilon \in (0,1)$ or $\beta > 0$ is fixed, and the learning rate or step sizes are chosen appropriately, then $\forall a \in \mathcal{A}$ and reachable states s, $Q_{\ell,t}(s,a) \to Q^*(s,a)$.*

Proof. Any reachable state is going to be visit i.o. and each action within each state will be tried i.o. □

One thing worth commenting here is that if we let $\epsilon \downarrow 0$ or $\beta \downarrow 0$, the above two algorithms can be shown to be GLIE.

Definition 4. *A learning policy is called GLIE (Greedy in the Limit with Infinite Exploration) if it satisfies the following two properties:*

1. *If a state is visited infinitely often, then each action in that state is chosen infinitely often (with probability 1).*
2. *In the limit (as $t \to \infty$), the learning policy is greedy with respect to the learned Q-function (with probability 1).*

The first condition requires exploration to proceed indefinitely, while the second requires that its magnitude decays in time. In practice either $\epsilon \downarrow 0$ or $\beta \downarrow 0$, this is to ensure that this algorithm reaches to the optimal behavioral and does not explore more afterwards. Note that for Boltzmann exploration, high temperature (β) means to choose action close to equiprobability (exploration); low temperature (β) means increasing bias towards best valued action (exploitation).

5 SARSA-state action reward state action algorithm

Next, we will move on to SARSA, which falls into the category of the "**on-policy**" schemes. This algorithm updates based on the current state-action pair and the next state-action pair.

Algorithm 3 SARSA algorithm

Initialize with some $Q_{0,0}$
for ℓ=0,1,2,$\cdots$ **do**
  sample $s_0 \sim \rho$
  select a_0
  for t=0,1,2,$\cdots,\tau-1$ **do**
    observe $s_{t+1} \sim \mathcal{P}_{s_t,a_t}$
    observe $s_{t+1} \sim \mathcal{R}_{s_t,a_t.s_{t+1}}$
    select a_{t+1}
    $$Q_{\ell,t+1}(s,a) = \begin{cases} (1-\gamma_\ell)Q_{\ell,t}(s,a) + \gamma_\ell(r_{t+1} + Q_{\ell,t}(s_{t+1},a_{t+1})), & \text{if } (s,a) = (s_t,a_t) \\ Q_{\ell,t}(s,a), & \text{otherwise} \end{cases}$$
  end for
  $Q_{\ell+1,0} = Q_{\ell,\tau}$
end for

If actions are selected according to some dithering strategy with a greedy policy in the limit, then under some conditions on the annealing rate, one can show that the policy converges to the optimal policy.

What we have discussed so far are all incremental update algorithms. The motivation for doing in the incremental update is intuitively to mimic how the brain works. Also, from an engineering perspective, incremental updates are easy to deal with. The opposite of the incremental algorithm is called batch algorithm. We will end our discussion with the batch algorithm today.

6 Sample-based value iteration

In sample-based value iteration we make use of previous observations. Lets define the data set with all the observations seen so far as

$$D = \{(s_k, a_k, r_k, s'_k) : k = 1, \cdots, K\}$$

The data set for a particular state-action pair will be

$$D_{s,a} = \{(\bar{s}, \bar{a}, \bar{r}, \bar{s'}) \in D : (\bar{s}, \bar{a}) = (s,a)\}$$

whose cardinality is $K_{s,a} = |D_{s,a}|$

We can define an approximate Bellman operator based in data $D_{s,a}$ as

$$(\hat{F}Q)(s,a) = \frac{1}{K_{s,a}} \sum_{(s,a,r,s') \in D_{s,a}} (r + \max_{a' \in A} Q(s',a'))$$

Then Approximate Value Iteration would be described by

$$Q(s,a) := \frac{1}{K_{s,a} + \bar{K}} (K_{s,a}(\hat{F}Q)(s,a) + \bar{K}\bar{Q}(s,a))$$

where $\bar{K}$ is a regularization parameter and $\bar{Q}(s,a)$ is the center of $Q(s,a)$, used when we have few observations.