Reinforcement Learning **Spring Semester, 2018/9**

Lecture 7: April 10, 2019

Lecturer: Yishay Mansour *Scribe: ym*

In this lecture we will continue looking at model-free learning. In the previous lecture we presented: (1) Q-learning, which is an off-policy learning algorithm that learns the optimal Q^* function, (2) `SARSA` which is an on-policy variant of Q-learning where we need to select actions, and (3) Monte-Carlo which is a model-free algorithm to learn the value function from episodes.

In this lecture we will look on temporal differences methods, which works in an online fashion. We will start with $TD(0)$ which uses the most recent observations for the updates, we will continue with methods that allow for a longer look-ahead, and then continue with $TD(\lambda)$ which averages multiple look-ahead estimations.

In general, temporal differences (TD) methods, learn directly from experience, and therefore are model-free methods. Unlike Monte-Carlo algorithms, they will use incomplete episodes for the updates, and they are not restricted to episodic MDPs. The TD methods update their estimates given the current observation and in that direction, similar in spirit to Q-learning and SARSA.

7.1 TD(0)

Fix a policy π. The goal is to learn the value function $V^\pi(s)$ for every $s \in S$. (The same goal as Monte-Carlo learning.) The methods will maintain an estimate $V_t(s)$ for $V^\pi(s)$ and use it for the updates. Unlike Monte-Carlo, there will be an interaction between the different estimates.

As a starting point, we can recall the *value iteration* algorithm.

$$V_{t+1}(s) = E^\pi[r(s,a) + \gamma V_t(s')]$$

We have shown that value iteration converges, namely $V_t \to V^\pi$.

Assume we sample (s_t, a_t, r_t, s_{t+1}). Assume that $\pi \in SD$, namely a stationary deterministic policy. Then,

$$E^\pi[\hat{V}_t(s_t)] = E^\pi[r_t + \gamma \hat{V}_t(s_{t+1})] = E^\pi[r(s,a) + \gamma \hat{V}_t(s') | s = s_t, a = \pi(s)]$$

The $TD(0)$ will do an update in the direction of the sample, namely, $[r_t + \gamma \hat{V}_t(s_{t+1})]$.

$TD(0)$ Algorithm

– Initialize $\hat{V}(s)$ arbitrarily.
– Update using (s_t, a_t, r_t, s_{t+1}):

$$\hat{V}(s_t) = \hat{V}(s_t) + \alpha_t(s_t, a_t)[r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t)]$$

where $\alpha_t(s, a)$ is the step size for (s, a) at time t.

We define the *temporal difference* to be

$$\Delta_t = r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t)$$

The $TD(0)$ update becomes:

$$\hat{V}(s_t) = \hat{V}(s_t) + \alpha_t(s_t, a_t)\Delta_t$$

In class we gave as a motivating example the driving example from [1]. (See the slides.)

We would like to compare the $TD(0)$ and the Monte-Carlo (MC) algorithms. Here is a simple example with four states $S = \{A, B, C, D\}$ where $\{C, D\}$ are terminal states and in $\{A, B\}$ there is one action (essentially, the policy selects a unique action). Assume we observe eight episodes. One episode is $(A, 0, B, 0, C)$, one episode $(B, 0, C)$, and six episodes $(B, 1, D)$. We would like to estimate the value function of the non-terminal states. For $V(B)$ both $TD(0)$ and MC will give $6/8 = 0.75$. The interesting question would be: what is the estimate for A? MC will average only the trajectories that include A and will get 0 (only one trajectory which gives 0 reward). The $TD(0)$ will consider the value from B as well, and will give an estimate of 0.75. (Assume that the $TD(0)$ continuously updates using the same episodes until it converges.)

We would like to better understand the above example. For the example the empirical MDP will have a transition from A to B, with probability 1 and reward 0, from B we will have a transition to C with probability 0.25 and reward 0 and a transition to D with probability 0.75 and reward 1. (See, Figure 7.1.) The value of A in the empirical model is 0.75. In this case the empirical model agrees with the $TD(0)$ estimate, we show that this holds in general.

Let the empirical model of a sample be define as follows. Let $n(s, a)$ be the number of times (s, a) appears in the sample. Given a sample (s_t, a_r, r_t, s_{t+1}) for

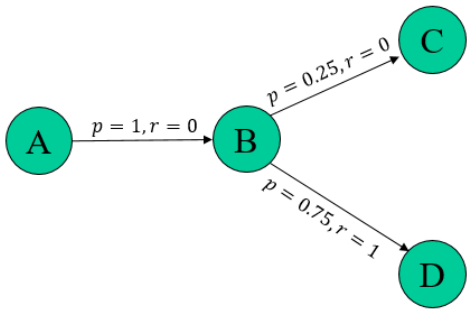

Figure 7.1: The situated agent

$1 \leq t \leq T$, we define the empirical model as follows. The rewards $\hat{r}(s,a)$ are the average rewards of (s,a), i.e., $\hat{r}(s,a) = \frac{\sum_{t:s_t=s,a_t=a} r_t}{n(s,a)}$ and $\hat{p}(s'|s,a) = \frac{n(s,a,s')}{n(s,a)}$, where $n(s,a,s')$ is the number of samples that have $s_{t+1} = s'$ when $s_t = s$ and $a_t = a$.

The main theorem would state that the value of π on the empirical model is identical to that of $TD(0)$ (running on the sample until convergence, namely, continuously sampling uniformly $t \in [1,T]$ and using (s_t, a_r, r_t, s_{t+1}) for the $TD(0)$ update).

Theorem 7.1. *Let V^π_{TD} be the estimated value function of π when we run $TD(0)$ until convergence. let V^π_{EM} be the value function of π on the empirical model. Then $V^\pi_{TD} = V^\pi_{EM}$.*

Proof sketch. The update of $TD(0)$ is $\hat{V}(s_t) = \hat{V}(s_t) + \alpha_t(s_t,a_t)\Delta_t$, where $\Delta_t = r_t + \gamma\hat{V}(s_{t+1}) - \hat{V}(s_t)$. At convergence we have $E[\Delta_t] = 0$ and hence,

$$\hat{V}(s) = \frac{1}{n(s,a)} \sum_{s_{t+1}:s_t=s,a_t=a} r_t + \gamma\hat{V}(s_{t+1}) = \hat{r}(s,a) + \gamma E_{s'\sim\hat{p}(\cdot|s,a)}[\hat{V}(s')]$$

where $a = \pi(s)$. □

The proof of the convergence of $TD(0)$ is very similar to that of Q-learning. We will show that $TD(0)$ is an instance of the stochastic approximation algorithm, as presented in previous lecture, and the convergence proof will follow from this.

Theorem 7.2 (Convergence $TD(0)$)**.** *If the step size has the properties that for every (s,a) we have $\sum_t \alpha_t(s,a) = \infty$ and $\sum_t \alpha_t^2(s,a) = O(1)$, then $\hat{V}$ converges to V^π, with probability 1.*

We will show the convergence using the general theorem for stochastic approximation iterative algorithm (which we saw in the previous lecture).

We first define a linear operator for the policy π,

$$(Hv)(s) = r(s, \pi(s)) + \gamma \sum_{s'} p(s'|s, \pi(s))v(s')$$

This is a contracting operator

$$\begin{aligned}\|Hv_1 - Hv_2\|_\infty &= \gamma \max_s |\sum_{s'} p(s'|s, \pi(s))(v_1(s') - v_2(s'))| \\ &\leq \gamma \max_{s'} |v_1(s') - v_2(s')| \\ &\leq \gamma \|v_1 - v_2\|_\infty\end{aligned}$$

This implies that H is γ-contracting.

We now would like to re-write the $TD(0)$ update to resemble the stochastic approximation iterative algorithm. The $TD(0)$ update is,

$$V_{t+1}(s_t) = (1 - \alpha_t)V_t + \alpha_t \Phi_t$$

where $\Phi_t = r_t + \gamma V_t(s_{t+1})$. We would like to consider the expected value of Φ_t. Clearly, $E[r_t] = r(s_t, \pi(s_t))$ and $s_{t+1} \sim p(\cdot|s_t, a_t)$. This implies that $E[\Phi_t] = (HV_t)(s_t)$. Therefore, we can define the noise term w_t as follows,

$$w_t(s_t) = [r_t + \gamma V_t(s_{t+1})] - (HV_t)(s_t) \ ,$$

and have $E[w_t] = 0$. We can bound $|w_t| \leq V_{max} = \frac{R_{max}}{1-\gamma}$, since the value function is bounded by V_{max}.

Returning to $TD(0)$, we can write

$$V_{t+1}(s_t) = (1 - \alpha_t)V_t + \alpha_t[(HV_t)(s_t) + w_t(s_t)]$$

The requirement of the step size appears in the statement of the theorem (and so holds). The noise w_t has both $E[w_t] = 0$ and $|w_t| \leq V_{max}$. And the operator H is contracting with a fix-point V^π. Therefore, we established Theorem 7.2.

When we compare $TD(0)$, to MC and both to Dynamic Programming (DP) we can view it as different ways of computing the value function.

MC We have $V^\pi(s) = E[R_t|s_t = s]$. In MC we observe the return, R_t, of episodes, and their mean is what we like to estimate.

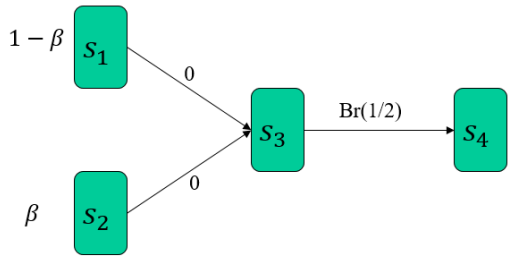

Figure 7.2: Markov Reward Chain

$TD(0)$ We have $V^\pi(s) = E[r_t + \gamma V^\pi(s_{t+1})|s_t = s]$. In $TD(0)$ we observe samples of r_t, we use our estimate for $V^\pi(s_{t+1})$ and the expectation is what we like to estimate (assuming our estimates converge).

DP We have $V^\pi(s) = \sum_a \pi(a|s)[r(s,a) + \gamma \sum_{s'} p(s'|s,a)V^\pi(s')]$. In DP we have the entire model, and use it to compute expectations.

We can see the difference between $TD(0)$ and MC in the Markov Chain in Figure 7.2. To get an approximation of state s_2, i.e., $|\hat{V}(s_2) - \frac{1}{2}| \approx \epsilon$. The Monte-Carlo will require $O(1/(\beta\epsilon^2))$ episodes (out of which only $O(1/\epsilon^2)$ start at s_2) and the $TD(0)$ will require only $O(1/\epsilon^2 + 1/\beta)$ since the estimate of s_3 will converge after $1/\epsilon^2$ episodes which start from s_1.

7.2 TD: Multiple look-ahead

The $TD(0)$ uses only the current reward and state. Given (s_t, a_t, r_t, s_{t+1}) it updates $\Delta_t = R_t^{(1)}(s_t) - \hat{V}(s_t)$ where $R_t^{(1)}(s_t) = r_t + \gamma\hat{V}(s_{t+1})$. We can also consider a two step look-ahead as follows. Given $(s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, s_{t+2})$ we can update using $\Delta_t^{(2)} = R_t^{(2)}(s_t) - \hat{V}(s_t)$ where $R_t^{(2)}(s_t) = r_t + \gamma r_{t+1} + \gamma^2\hat{V}(s_{t+2})$. Using the same logic, we have that this is a temporal difference that uses a two time steps.

We can generalize this to any n-step look-ahead and define $R_t^{(n)}(s_t) = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n\hat{V}(s_{t+n})$ and updates $\Delta_t^{(n)} = R_t^{(n)}(s_t) - \hat{V}(s_t)$.

We can relate the $\Delta_t^{(n)}$ to the Δ_t as follows:

$$\Delta_t^{(n)} = \sum_{i=0}^{n-1} \gamma^i \Delta_{t+i}$$

This follows since

$$\sum_{i=0}^{n-1} \Delta_{t+i} = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \sum_{i=0}^{n-1} \gamma^{i+1} \hat{V}(s_{t+i+1}) - \sum_{i=0}^{n-1} \gamma^i \hat{V}(s_{t+i}) = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n \hat{V}(s_{t+n}) - \hat{V}(s_t)$$

Using the n-step look-ahead we have $\hat{V}(s_t) = \hat{V}(s_t) + \alpha_t \Delta_t^{(n)}$ where $\Delta_t^{(n)} = R_t^{(n)}(s_t) - \hat{V}(s_t)$. Note that the operator $R_t^{(n)}$ is γ^n-contracting, namely

$$\|R_t^{(n)}(V_1) - R_t^{(n)}(V_2)\|_\infty \leq \gamma^n \|V_1 - V_2\|_\infty$$

We can use any parameter n for the n-step look-ahead. If the episode ends before step n we can pad it with rewards zero. This implies that for $n = \infty$ we have that n-step look-ahead is simply the Monte-Carlo estimate. However, we need to select some parameter n. An alternative idea is to simply average over the possible parameters n. One simple way to average is to use exponential averaging with a parameter $\lambda \in (0, 1)$. This implies that the weight of parameter n is $(1 - \lambda)\lambda^{n-1}$.

This leads us to the $TD(\lambda)$ update:

$$\hat{V}(s_t) = \hat{V}(s_t) + \alpha_t (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} \Delta_t^{(n)}$$

Remark: While both γ and λ are used to generate exponential decaying values, their goal is very different. The discount parameter γ defines the objective of the MDP, the goal that we like to maximize. The exponential averaging parameter λ is used to average over the different look-ahead parameters, and is selected to optimize the convergence.

In the slides there is an example of a random walk and its convergence taken from [1].

The above describes the forward view of $TD(\lambda)$, where we average over future rewards. If we will try to implement it in a strict way this will lead us to wait until the end of the episode, since we will need to first observe all the rewards. Fortunately, there is an equivalent form of the $TD(\lambda)$ which uses a *backward view*. The backward view updates at each time step, using an incomplete information. At the end of the episode, the updates of the forward and backward updates will be the same.

The basic idea of the backward view is the following. Fix a time t and a state $s = s_t$. We have at time t a temporal difference $\Delta_t = r_t + \gamma V_t(s_{t+1}) - V_t(s_t)$. Consider how this Δ_t affects all the previous times $\tau < t$ where $s_\tau = s = s_t$. The influence is exactly $(\gamma\lambda)^{t-\tau}\Delta_t$. This implies that for every such τ we can do the desired update,

however, we can aggregate all those updates to a single update. Let,

$$e_t(s) = \sum_{\tau \leq t} (\gamma\lambda)^{t-\tau} = \sum_{\tau=1}^{t} (\gamma\lambda)^{t-\tau} I(s_\tau = s)$$

The above $e_t(s)$ defines the *eligibility trace* and we can compute it online using

$$e_t(s) = \gamma\lambda e_{t-1}(s) + I(s = s_t)$$

which result in the update

$$\hat{V}_{t+1}(s) = \hat{V}_t(s) + \alpha_t e_t(s) \Delta_t$$

Note that for $TD(0)$ we have that $\lambda = 0$ and the eligibility trace becomes $e_t(s) = I(s = s_t)$. This implies that we update only s_t and $\hat{V}_{t+1}(s_t) = \hat{V}_t(s_t) + \alpha_t \Delta_t$.

$TD(\lambda)$ **algorithm**

- Initialization: Set $\hat{V}(s) = 0$ (or any other value), and $e_0(s) = 0$.
- Update: observe (s_t, a_t, r_t, s_{t+1}) and set

$$\begin{aligned} \Delta_t &= r_t + \gamma \hat{V}_t(s_{t+1}) - \hat{V}(s_t) \\ e_t(s) &= \gamma\lambda e_{t-1}(s) + I(s = s_t) \\ \hat{V}_{t+1}(s) &= \hat{V}_t(s) + \alpha_t e_t(s) \Delta_t \end{aligned}$$

To summarize, the benefit of $TD(\lambda)$ is that it interpolates between $TD(0)$ and Monte-carlo updates, and many times achieves superior performance to both. One can show the equivalence of the forward and backward $TD(\lambda)$ updates (see [1]). Also, $TD(\lambda)$ can be written as a stochastic approximation iterative algorithm, and one can derive its convergence.

7.2.1 $SARSA(\lambda)$

We can use the idea of eligibility traces also in other algorithms, such as SARSA. Recall that given $(s_t, a_t, r_t, s_{t+1}, a_{t+1})$ the update of SARSA is

$$r_t + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)$$

Similarly, we can define an n-step look-ahead $q_t^{(n)} = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n Q_t(s_{t+n}, a_{t+n})$ and set $Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(q_t^{(n)} - Q_t(s_t, a_t))$.

We can now define $SARSA(\lambda)$ using exponential averaging with parameter λ. Namely, we define $q_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}q_t^{(n)}$. This makes the forward view of $SARSA(\lambda)$ to be $Q_{t+1}(s_t,a_t) = Q_t(s_t,a_t) + \alpha_t(q_t^\lambda - Q_t(s_t,a_t))$.

Similar to $TD(\lambda)$, we can define a backward view using eligibility traces:

$$\begin{aligned} e_0(s,a) &= 0 \\ e_t(s,a) &= \gamma\lambda e_{t-1}(s,a) + I(s=s_t, a=a_t) \end{aligned}$$

For the update we have

$$\begin{aligned} \Delta_t &= r_t + \gamma Q_t(s_{t+1},a_{t+1}) - Q_t(s_t,a_t) \\ Q_{t+1}(s,a) &= Q_t(s,a) + \alpha_t e_t(s,a)\Delta_t \end{aligned}$$

7.3 Importance Sampling

Importance sapling is a simple general technique to estimate the mean with respect to a given distribution, while sampling from a different distribution. To be specific, let Q be the sampling distribution and P the evaluation distribution. The basic idea is the following

$$E_{x\sim P}[f(x)] = \sum_x P(x)f(x) = \sum_x Q(x)\frac{P(x)}{Q(x)}f(x) = E_{x\sim Q}[\frac{P(x)}{Q(x)}f(x)]$$

This implies that given a sample $\{x_1,\ldots,x_m\}$ from Q, we can estimate $E_{x\sim P}[f(x)]$ using $\sum_{i=1}^m \frac{P(x_i)}{Q(x_i)}f(x_i)$.

The importance sampling gives an unbiased estimator, but the variance of the estimator might be huge, since it depends on $P(x)/Q(x)$.

We would like to apply the idea of importance sampling to learning in MDPs. Assume that there is a policy π that selects the actions, and there is a policy ρ that we would like to evaluate. For the importance sampling, given a trajectory, we need to take the ratio of probabilities under ρ and π.

$$\frac{\rho(s_1,a_1,r_1,\ldots,s_T,a_T,r_T,s_{T+1})}{\pi(s_1,a_1,r_1,\ldots,s_T,a_T,r_T,s_{T+1})} = \prod_{t=1}^{T}\frac{\rho(a_t|s_t)}{\pi(a_t|s_t)}$$

where the eq7uality follows since the reward and transition probabilities are identical, and cancel.

For Monte-Carlo, the estimates would be

$$G^{\rho/\pi} = \prod_{t=1}^{T} \frac{\rho(a_t|s_t)}{\pi(a_t|s_t)} (\sum_{t=1}^{T} r_t)$$

and we have

$$\widehat{V}^{\rho}(s_1) = \widehat{V}^{\rho}(s_1) + \alpha(G^{\rho/\pi} - \widehat{V}^{\rho}(s_1))$$

This updates might be huge, since we are multiplying many small numbers.

For the $TD(0)$ the updates will be

$$\Delta_t^{\rho/\pi} = \frac{\rho(a_t|s_t)}{\pi(a_t|s_t)} r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t)$$

and we have

$$\widehat{V}^{\rho}(s_1) = \widehat{V}^{\rho}(s_1) + \alpha(\Delta_t^{\rho/\pi} - \widehat{V}^{\rho}(s_1))$$

This update is much more stable, since we have only one factor multiplying the observed reward.

7.4 Actor-critic methodology

Actor-critic gives a general methodology of building reinforcement learning algorithms. It is composed from an actor, that selects the actions, ad a critic, that learns the value function. The actor observes the current state, and the value function, and selects an action. The critic, observes the current state (and action) and reward and outputs a value function. See Figure 7.3.

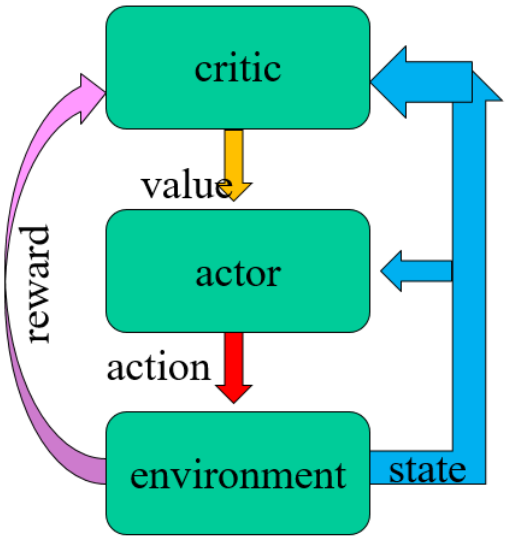

Figure 7.3: The situated agent

Bibliography

[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998.