Computational Learning Theory **Fall Semester, 1999/00**

Lecture 8: December 16

Lecturer: Yishay Mansour *Scribe: Alexander Rusinov, Moti Alperovitch*

8.1 Evaluation Policy Reward using MC and TD(0)

8.1.1 Another way to look on MC algorithm

In lecture 7 we discussed Monte-Carlo (MC) method for evaluation policy reward.
This method performs number of experiments and uses the average to evaluate policy reward.

Another way to express the evaluation is the following:
$V_{n+1}(s) = V_n(s) + \alpha(R_n(s) - V_n(s))$
where R_n is total reward of n-th run starting first visit in s (given s was visited in this run).
Note that $E[R_n(s)] = V^{\pi}(s)$

We can rewrite this formula ,as follows,
$V_{n+1}(s) = (1-\alpha)V_n(s) + \alpha[V_n^{\pi}(s) + (R_n(s) - V_n^{\pi}(s))]$

where: $HV_n = V_{n+1}^{\pi}$ - for some nonlinear operator H ,and $W_n = R_n - V_n^{\pi}$ is "noise" and $E(W_n) = 0^{\cdot}$

Recall the operator $L_{\pi}\vec{V} = \vec{R_{\pi}} + \lambda P_{\pi}\vec{V}$, that was introduced to compute the return of policy. We've already shown that L_{π} is a contracting operator.

8.1.2 Temporal Difference and TD(0) algorithm

We can write down algorithm according to the defined operator:

$V_{n+1}(s) \leftarrow (1-\alpha_n)V_n(s) + \alpha_n[r_{\pi}(s) + \lambda E_s[V(s')]]$

Instead of computing expectation E_s we sample the next state s' and derive

$V_{n+1}(s) \leftarrow (1-\alpha_n)V_n(s) + \alpha_n[r_{\pi}(s) + \lambda V_n(s')]$ or equivalently,
$V_{n+1}(s) \leftarrow V_n(s) + \alpha_n[r_{\pi}(s) + \lambda V_n(s') - V_n(s)]$

We'll call $r_\pi(s) + \lambda V_n(s') - V_n(s)$ - temporal difference (TD) because this expression is difference between reward received in current run $r_\pi(s)+\lambda V_n(s')$ and expected reward $V_n(s)$˙

This algorithm is called TD(0).

It is easy to see that $E(r_\pi(s) + \lambda V_n(s')) = E(V_n(s))$ and $E(TD) = 0$

The following theorem is used to show convergence of the algorithm:

Theorem 8.1 *Consider the iterative algorithm $r_{t+1}(i) \leftarrow (1 - \alpha_t(i))r_t(i) + \alpha_t(i)[(Hr_t)(i) + w_t(i)]$ such that*
1) $E(w_t(i)|F_t) = 0$ and $E(w_t^2(i)|F_t) = A + B||r_t||^2$ for some constants A and B
2) the step size α_t has the property that $\sum \alpha_t = \infty$ and $\sum \alpha_t^2 < \infty$;
3) H is a contracting mapping.
Then r converges to r^* with probability one, when $r^* = Hr^*$

Recall that convergence with probability one implies that sequence A_n has the property that $\lim_{N\to\infty} sup_{n\geq N}|A_N - A| = 0$˙The above theorem implies that TD(0) converges with probability one to the correct value function (same as MC algorithm).

8.1.3 Online Versus Off-Line Updates

The basic advantage of TD(0) is in possibility to perform online updates based on information received up to current point. In TD(0) all we need to know to perform update is the next state.

Here is an example of TD(0)-like algorithm run :
An employee estimates time he need to get home from work.

1. State: Leaving for home ; Elapsed time : 0 ; Estimated time to goal: 30 ; Total estimation: 30
2. State: Got into the car, It's raining ; Elapsed time : 5 ; Estimated time to goal: 35 ; Total estimation: 40
3. State: Getting out of highway ; Elapsed time : 20 ; Estimated time to goal: 15 ; Total estimation: 35
4. State: There is big truck ahead; Elapsed time : 30 ; Estimated time to goal: 10 ; Total estimation: 40

5. State: Got to the right street ; Elapsed time : 40 ; Estimated time to goal: 3 ; Total estimation: 43
6. State: Got home ; Elapsed time : 43 ; Estimated time to goal: 0 ; Total estimation: 43

For MC-like algorithm we need to finish the run for making updates. For TD(0) updates are made online (hopefully in right direction) and in MC all updates are made only after a run is finished.

8.1.4 Differences between TD(0) and MC

If we run TD(0) and MC for long enough time (and gives them enough data) both algorithms will converge to the correct value function. However on the finite input , algorithms can give different results.

For understanding basic difference between MC and TD(0) we'll study following example: Let us assume that we have run series of runs and got following results:

1. (A,0) (B,0)
2. (B,1)
3. (B,1)
4. (B,1)
5. (B,1)
6. (B,1)
7. (B,1)
8. (B,0)

We can create infinite sample from such a finite sample as follows. Each time we need to sample a run, we choose a random run out of the eight runs.

We want to estimate V(A) and V(B) using this finite sample.
For V(B) we have 8 runs, 6 give return 1 and 2 of them give return 0. Our estimate for V(B)

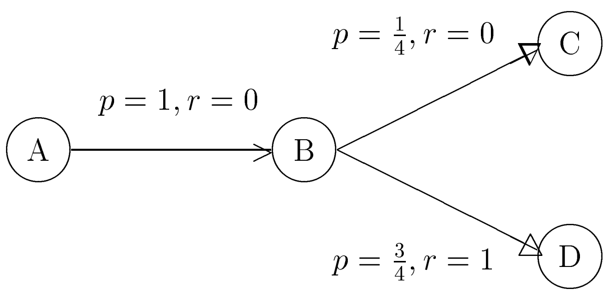

Figure 8.1: Empirical model.

is $\frac{3}{4}$.
How should we estimate V(A)? There are two reasonable options.

1. We have one run visiting state A with return 0, so V(A) estimation is 0.
2. We have one run visiting state A and this run moves from state A to state B. So we can assume that we always move from state A to state B. This implies that estimation is V(A) = V(B) = $\frac{3}{4}$.

Clearly first approach corresponds to MC. Now we show that TD(0) implements second approach. It means that TD(0) evaluation is in fact done using Markovian model produced by inspecting input : for any state s an action a the next state distribution is identical to the one observed in the sample.

Claim 8.2 *Given finite sample $T_1 ... T_l$. If we run TD(0) until it converges (sampling every time randomly one of the runs $T_1 ... T_l$) then estimation $\widehat{V}^{\pi}$ equals to return of policy π in empirical model for $T_1 ... T_l$.*

Definition
Given runs $T_1 ... T_l$, the empirical model is defined as follows:
$\forall s \in S, a \in A$: $\widehat{p}(s'|s,a) = \frac{\#(s,a,r,s')}{\#(s,a)}$ and $\widehat{r}(s,a) = Avg(r(s,a))$
where $\#(s,a)$ is the number of times in $T_1 ... T_l$ we did action a in state s and $\#(s,a,r,s')$ is the number of times when after doing action a we reached state s'.

Explanation : It is easy to see from the definition of TD(0) that based are impacted only on successive pairs of states $\#(s, a, r, s')^{\cdot}$

$$V_{n+1}(s) \leftarrow V_n(s) + \alpha_n[\widehat{r} + \lambda V_n(s') - V_n(s)]$$

Number of updates for state s' is distributed according to $\widehat{p}$ and so:

$$V_{n+1}(s) \leftarrow V_n(s) + \alpha_n[\widehat{r} + \lambda E_{\widehat{p} \sim s'}[V_n(s')]]$$

We received equations for model with $\widehat{p}$ and $\widehat{r}$ which is exactly the empirical model.

Conclusion. Here is conceptual difference between MC and TD(0):
MC considers only information about initial state of the run (received at the end of the run) and TD(0) assumes that there is Markovian model and makes use of this assumption.

8.2 TD(γ) algorithm

8.2.1 TD(γ) as generalization of MC and TD(0) methods

MC uses overall return of the run to perform update, i.e., $R_t = \sum_{i=0}^{T} \lambda^i r_{t+i}$
TD(0) uses only immediate reward to make update and its update has the form,
$R_t^{(1)} = r_t + \lambda V_t(s_{t+1})$

We can think about other options between those two extremes. For example we can use two immediate reward values , i.e. , $R_t^{(2)} = r_t + \lambda r_{t+1} + \lambda^2 V_t(s_{t+2})$

Generally - if n values are in use, we have, $R_t^{(n)} = r_t + \lambda r_{t+1} + ... + \lambda^{n-1} r_{n-1} + \lambda^n V_t(S_n)$
Note: If n is more than number of steps to the end of the run, we set all the rewards after the run end to 0.

The update is defined as $\triangle V_t(s_t) = \alpha[R_t^{(n)} - V_t(s_t)]$

Update can be performed:

1. online: $V_{t+1}(s_t) \leftarrow V_t(s_t) + \triangle V_t(s_t)$
2. off-line: $V(s) \leftarrow V(s) + \sum \triangle V_t(s)$

It is easy to see that operator $R_t^{(n)}$ is λ^n contracting.
$[\|R_t^{(n)}(V^{(1)}) - R_t^{(n)}(V^{(2)})\| = \lambda^n \|V^{(1)} - V^{(2)}\|]$
So we expect to get improvement and reduce error while making iterations.

The central remaining question is choice of n . One way to do it is to use weighed average of all possible values.

The method $TD(\gamma)$ gives a geometric distribution with parameter γ over $R_t^{(n)}$, i.e. weight of $R_t^{(n)}$ is $(1-\gamma)\gamma^{n-1}$.

We define R_t^γ to be $R_t^\gamma = (1-\gamma)\sum_{n=1}^{\infty}\gamma^{n-1}R_t^{(n)}$

For finite run this expression can be rewritten as :
$R_t^\gamma = (1-\gamma)\sum_{n=1}^{T-1}\gamma^{n-1}R_t^{(n)} + \gamma^T R_t$

1. If $\gamma = 0$ then weight of $R_t^{(1)}$ is 1 and weights of any other $R_t^{(n)}$ equals 0. This reduces TD(0), which we introduced before.

2. For $\gamma = 1$ we'll get $R_t^{(1)} = R_t$ (total reward) .

8.2.2 Forward Updates Versus Backward Updates

In all definitions of $TD(\gamma)$ we looks on updates occurring in forward direction in time (looking to the "future") like in MC method. We want to define scheme for online updates and so we need to find the way to look on updates in backward direction ("past"). We like that at the end of the run, updates would be the same for both schemes.

For implementation of such method, an additional variable which holds weight of a state in a run will be provided for every run. Every step we multiply all those variables (called eligibility trace) by $\gamma\lambda$ and for current state we add 1. Formally,

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

Every time we visit a state , its weight is going up and for states not visited theirs weight is going down. Let us define: $\delta_t = r_{t+1} + \lambda V_t(s_{t+1}) - V_t(s_t)^{\cdot}$
Now the update would be, $\Delta V_t(s) = \alpha\delta_t e_t(s)^{\cdot}$

Algorithm

The algorithm is as follows:

1. Define an arbitrary $V(s)$, $\forall s \in S$ set $e(s) = 0$
2. Repeat for every run : Set initial state s_0
3. Repeat for every step of the run:

```
    · Choose action according to policy $\pi : a \leftarrow \pi(s)$
    · Perform action $a$ , receive next state $s'$ and immediate reward $r_t$
    ·$\delta_t \leftarrow r_t + \lambda V_t(s') - V_t(s)$
    ·$e_{t+1}(s) \leftarrow \gamma\lambda e_t(s) + 1$
    ·$\forall f \in S$ set $e_{t+1}(f) \leftarrow \gamma\lambda e_t(f)$ for $f \neq s$ and $V_{t+1}(f) \leftarrow V_t(f) + \alpha\delta e_t(f)$
· repeat steps 2,3 until final state is reached (for every run)
```

Using this scheme, we update previous values according to new immediate reward and according to state we have reached. For more distant states weight of the update is smaller and for more close states the weight is larger.

Let us examine different values of γ .

· For $\gamma = 0$: $e(s) = 1$ iff $s = s_t$ and $e(s) = 0$ otherwise
So, we will update only current state according to received reward and we once again have the TD(0) scheme.

· For $\gamma = 1$: $e(s)$ is contribution of this step to state s, i.e. if s is visited in the run in steps $i_1, ..., i_l$ then $e_t(s) = \sum_{j=1}^{l} \lambda^{l-i_j}$
where λ^{l-i_j} is contribution of current step to step i_j

Although TD(1) resembles MC , there is a minor difference : MC waits for the end of the run to perform updates while TD(1) performs updates online. By the end of the run, both methods give the same value function , but functions computed by TD(1) will be hopefully more accurate during the run, since they are updated online.

8.2.3 Equality of schemes using backward and forward updates

Now we will show that "looking forward" and "looking backward" schemes are equal (i.e. will give same results).
Let us say:

$\Delta V_t^\gamma(s_t) = \alpha[R_t^\gamma - V_t(s_t)]$ for the forward scheme

$\Delta V_t^{TD}(s) = \alpha\delta_t e_t(s)$ for the backward scheme
We will show that

$\Sigma_{t=0}^{T-1}\Delta V_t^{TD}(s) = \Sigma_{t=0}^{T-1}\Delta V_t^\gamma(s_t)I(s,s_t)$
where,
$I(s,s_t)=1$ if $s=s_t$,and $I(s,s_t)=0$ if $s\neq s_t$

We can rewrite $e_t(s)$ as $e_t(s)=\Sigma_{k=0}^{t}(\gamma\lambda)^{t-k}I(s,s_k)$

We have for the left side of the equation:

$\Sigma_{t=0}^{T-1}\Delta V_t^{TD}(s) = \Sigma_{t=0}^{T-1}\alpha\delta_t\Sigma_{k=0}^{t}(\gamma\lambda)^{t-k}I(s,s_k)$

$= \Sigma_{k=0}^{T-1}\alpha\Sigma_{t=0}^{k}(\gamma\lambda)^{k-t}I(s,s_t)\delta_k$

$= \Sigma_{t=0}^{T}\alpha\Sigma_{k=0}^{T-1}(\gamma\lambda)^{k-t}I(s,s_t)\delta_k$

$= \Sigma_{t=0}^{T-1}\alpha I(s,s_t)\Sigma_{k=t}^{T-1}(\gamma\lambda)^{k-t}\delta_k$

And for the right side:

$\frac{1}{\alpha}\Delta_t^\gamma(s) = R_t^\gamma - V_t(S)$
$= -V_t(s_t) + (1-\gamma)\gamma^0[r_t+\gamma V_t(s_t)] + (1-\gamma)\gamma[r_t+\gamma r_{t+1}+\gamma^2 V_t(s_t)]+ \ldots$

$= -V_t(s_t) + \Sigma_{i=0}^{\infty}(\lambda\gamma)^i[r_{t+i}+\lambda(1-\gamma)V_t(s_{t+i+1})]$

$= -V_t(s_t) + \Sigma_{i=0}^{\infty}(\lambda\gamma)^i[r_{t+i}+\lambda V_t(s_{t+i+1}) - V_t(s_{t+i})]$

$= \Sigma_{k=t}^{\infty}(\lambda\gamma)^{k-t}\delta_k = \Sigma_{k=t}^{T-1}(\lambda\gamma)^{k-t}\delta_k$

So, we have :
$\Sigma_{t=0}^{T}\Delta V_t^{\gamma}(s_t)I(s,s_t) = \Sigma_{t=0}^{T-1}\alpha I(s,s_t)\Sigma_{k=t}^{T-1}(\lambda\gamma)^{k-t}\delta_k$

We have for right side the same expression we have already for the left side of the equation. This implies that both methods give same updates at the end of the run.