Lecture 9: December 23

Lecturer: Prof. Yishay Mansour *Scribe: Harrusi Shachar, Sozio Andrei,*

9.1 Evaluating One Policy With Another

Until now we discussed the case we have policy π and need to evlaute its value V^{π}.
Now we look at the case where we have two policies: π_1,π_2. We have samples of π_1 and we need to evaluate $V\pi_2$.

9.1.1 Importance Sampling

We have two sources $D_1(x)$ and $D_2(x)$ that produce differnt distributions. We compute expectation of a function F(x) on one source while sampling the other source. The expectation of F(x) with respect to distribution D is the sum of products of all values of X with the probability that D assigns that value. In our case:

$E_{D_2}[F(x)] = \sum D_2(X)F(x) = \sum D_1(X)(\frac{D_2(X)}{D_1(X)})F(x) = E_{D_1}[(\frac{D_2(X)}{D_1(X)})\mathrm{F(x)}]$

<u>EXAMPLE 1</u>
Input:

- $F(x) = k$.
- $D_1 = Prob[k] = \frac{1}{2^k}$.
- $D_2 = Prob[k] = \frac{3}{4}(\frac{1}{4})^{k-1}$.

Computation:

- We find expectancy $E_{D_2}[k]$ from samples of D_1.

  $E_{D_1}[k] = E_{D_2}[(\frac{D_1(k)}{D_2(k)})k] = E_{D_2}[(\frac{(\frac{1}{2})^k}{(\frac{3}{4})(\frac{1}{4})^{(k-1)}})k] = E_{D_2}[k\frac{4}{3}\frac{1}{2}2^{k-1}] = E_{D_2}[k\frac{2}{3}^k]$

- We check the equation by computing $E_{D_2}[F(x)]$.

  $E_{D_2}[k\frac{2}{3}^k] = \sum_k(\frac{1}{4})^{k-1}(\frac{3}{4})(\frac{2^k}{3})(k) = \sum_k(\frac{1}{2^k})k = E_{D_1}[k]$

- One of the problems in importance sampling is the variance.
  In this case.

  $E_{D_2}[(k\frac{2^k}{3})^2] = \sum_k (k^2)(\frac{2^{2k}}{9})(\frac{1}{4})^{k-1}\frac{3}{4} = \sum_k \frac{k^2}{3} = \infty$

9.1.2 Policy Sampling

The conclusion from the above equivalence is that if we can compute $\frac{D_2(X)}{D_1(X)}$ then we able to "transform" samples from distribution D_1 to samples in distribution D_2.
We produce $T \in \{S, A, R\}^* : T_1..T_m$ of policy π_1. each sample(T_j) is a run on the model using policy π_1, i.e. $T_j = s_1, a_1, r_1, s_2, a_2, r_2...$
The probability of generating T_j is actually a product of two independent probabilities: a policy depeneded probablity on actions and a model depended probability.

$Prob[T_j] = \prod_{i=1}^{|T_j|}(\pi_1(s_i, a_i) * Prob(s_{i+1}|s_i, a_i)) =$
$(\prod_{i=1}^{|T_j|}\Pi_1(s_i, a_i)) * (\prod_{i=1}^{|T_j|}Prob(s_{i+1}|s_i, a_i))$

We calculte the ratio of probabilites to have the same T_j on differnt policies:

$\frac{Prob_{\pi_2}[T_j]}{Prob_{\pi_1}[T_j]} = \frac{(\prod_{i=1}^{|T_j|}\pi_2(s_i,a_i))*(\prod_{i=1}^{|T_j|}Prob(s_{i+1}|s_i,a_i))}{(\prod_{i=1}^{|T_j|}\pi_1(s_i,a_i))*(\prod_{i=1}^{|T_j|}Prob(s_{i+1}|s_i,a_i))} = \prod_{i=1}^{|T_j|}\frac{\pi_2(s_i,a_i)}{\pi_1(s_i,a_i)}$

The important fact is that the ratio does <u>not</u> depened on the model, but only on the policies. Therefor we can compute it with out the model.
<u>EXAMPLE 2</u>

Input:

- policy Π_1 .
- policy Π_2 is determinstic.

Computation:

- because Π_2 is determinstic $\Pi_2(s_i, a_i) \in \{0, 1\}$
- Therefor the ratio is a product of X, $X \in \{0, \frac{1}{\Pi_1(s,a)}\}$
- If Π_1 is the random policy than we simply have a uniform distribution on the runs. Consistent with Π_2.

Problem of sampling

We can estimate policy π_2 from samples of another policy π_1. The problem is the variance can be big -infinte. (variance in example 1). Big variance can cause the error of the estimation grow.

conclusion:

1. To calculate the ratio we don't need any knowledge on the Model only about the two policies we use.
2. The ratio is $(\frac{D_2(X)}{D_1(X)})F(x)$ is the case of Importance Sampling.
3. 1+2 imply we can use samples from one policy to calculate samples on another policy.
4. conclusion 3 explain why Q-learning can work.
5. The Variance must be limited to avoid errors.

9.2 Q-learning and SARSA algorithms

In this section we descuss off-line and on-line algorithms to compute the optimal policy in case the exact model is not known.

9.2.1 Q-learning

Lets consider Value Iteration Algorithm(VI) from lecture 6.It described the non linear operator: L. In every iteration of the algorithm we operate L:
$V_{n+1} = LV_n$, and explicitaly:

$V_{n+1} = \max_{a \in A_s}\{r(s,a) + \lambda \sum_{s' \in S} P(s'|s,a)V_n(s')\}$.

Lets refine the equation a somewhat. We define new function Q regarding VI:

$Q^{n+1}(s,a) = r(s,a) + \lambda \sum_{s' \in S} P(s'|s,a)V_n(s')$.

Now the iteration of VI are: $V_n = \max_{a \in A_s}\{Q^n(s,a)\}$. Expressed in Q function terms only we have:

$Q^{n+1}(s,a) = r(s,a) + \lambda \sum_{s' \in S} P(s'|s,a) \max_{b \in A_s}\{Q^n(s',b)\}$.

We write the iteration with $\alpha - notation$.
(In lecture 7 we learned it converges the right value.)

$Q^{n+1}(s,a) = (1-\alpha)Q^n(s,a) + \alpha[r(s,a) + \lambda \sum_{s' \in S} P(s'|s,a) \max_{b \in A_s}\{Q^n(s',b)\}]$

Until now the iterations are equvivalent to VI. Instead of taking the excpetancy of the value of the next step we take a sample of the next step. We assume that we are in state s, we take action a, the next state s' is distributed by $P(s'|s,a)$. Finally we get

$Q^{n+1}(s,a) = (1-\alpha)Q^n(s,a) + \alpha[r(s,a) + \lambda \max_{b \in A_{s'}} Q^n(s',b)\}]$

```
Q-LEARNING
  begin
  Init: Q(s,a)
  for every run T
    Init s
    for every step in T
      choose action a (using policy Π)
      do action a and obtain: s' ∈ S ,r ∈ R
      Q(s,a) = Q(s,a) + α[r(s,a) + λ max_{b∈A_s'}{Q(s',b)} − Q(s,a)]
      s ← s'
      end for
    end for
  end Q-LEARNING
```

Figure 9.1: Algorithm for Q-LEARNING

.

remarks:

- if we choose Q to be optimal $Q = Q^*$ then $Q^*(s,a) = r(s,a) + \lambda E_{s'}[^{max}_{b} Q^*(s',b)]$.
- The algorithm Q-Learning is off-policy since Π we don't control the policy that porforms actions. In general an off-line algorithm doesn't control the actions it does.
- For on-policy we can give any action a small probability s.t. we reach all MDP. For on-policy we can hope to achieve rewards getting closer to optimal.

For on-policy π is $\epsilon - greedy$ regarding Q at the moment. Thus we get Sarsa algorithm

9.2.2 SARSA

SARSA is on-line algorithm. In this algorithm keep two states the current state and the next state.
Hence its name "S(s),A(a),R(r),S(s'),A(a')" is derive from the fact that we use the current state (s), current action (a), current reward (r), next state (s') and next action (a'). We update Q with the difference between the next value function in s' and the current value function.

SARSA
begin
Init: $Q(s,a)$
for every run T
  Init s
  choose action a for s using $\epsilon - greedy$ policy for Q
  for every step $\in T$
    do action a and get: $s' \in S$, $r \in R$
    choose action a' for s' (from Q).
    $Q(s,a) = Q(s,a) + \alpha[r(s,a) + \lambda(Q(s',a')) - Q(s,a)]$
    $s \leftarrow s'$
    $a \leftarrow a'$
    end for
  end for
end SARSA

Figure 9.2: Algorithm for SARSA

.

9.3 Convergence proof

The general idea is writing our algorithm as:

$Q \leftarrow (1-\alpha)Q + \alpha(HQ + w)$

where:

$(HQ)(s,a) = r(s,a) + \lambda \sum_{s'} P(s'|s,a)_b^{max} Q(s',b)$

We will show that H is a contracting mapping (operator):
$||HQ_1 - HQ_2|| = ||\lambda P[MQ_1 - MQ_2]||$
where $(MQ_1)(s) =_a^{max} Q_1(s,a)$. Note that $MQ_1 = V_1$. We have,
$||HQ_1 - HQ_2|| \leq \lambda||V_1 - V_2|| \leq \lambda||Q_1 - Q_2||$,
Therefore H is λ contracting.
The parameter w is "noise" in the sample whose expectation is 0. By using the contracting property of H we show convergence. For convergence we require:

- $Q^{n+1}(s,a) = (1-\alpha_n(s,a))Q^n(s,a) + \alpha_n(s,a)[r(s,a) + \lambda_b^{max} Q^n(s',b)]$
- Let $T^{s,a}$ be the time we are in state s and make action a
  - for $t \in T^{s,a}$ $\alpha_t(s,a) > 0$
  - and for $t \notin T^{s,a}$ $\alpha_t(s,a) = 0$
- F_t is history untill time t
- Thus the algorithm can be written as:

  $$Q^{n+1}(s,a) = (1-\alpha_n(s,a))Q^n(s,a) + \alpha_n(s,a)[(HQ^{(n)}(s,a) + w_n(s,a)]$$

  where: $w_n(s,a) = \lambda[_b^{max} Q^{(n)}(s',b) - \sum_{s''} P(s''|s,a)_c^{max} Q^{(n)}(s'',c)]$
- because s' is distributed according to $P(.|s,a)$ (the policy is stationary) then we have:
  $E[w_n(s,a)] = \lambda[E[_b^{max} Q^{(n)}(s',b)] - E[_c^{max} Q^{(n)}(s'',c)]] = 0$
  Therefore the expectation of "noise" is 0.

Theorem 9.1 *(A general theorem for Stochastic processes)*
Let: $r_{t+1}(i) = (1-\alpha_t(i))r_t(i) + \alpha_t(i)[(Hr_t)(i) + w_t(i)]$

Given that:

1. $E[w_t(i)|F_t] = 0$
2. $E[w_t^2(i)|F_t] \leq A + B||r_t||^2$ *for some A,B constants.*
3. *the steps* $\alpha_t(i)$ *are such that:* $\sum \alpha_t(i) = \infty$ *and* $\sum \alpha_t^2(i) < \infty$
4. *the mapping H (operator) is contracting*

Then $r_t \rightarrow r^*$ *with probability 1 .*
(where R^* *is the fixed point of* $Hr^* = r^*$*)*

In the case of Q-Learning it means that $HQ^* = Q^*$ and Q^* is the optimal value function, (this is the same operator as in Value Iteration). Thus we achieve by the convergence of Q-Learning the optimal policy value.

Theorem 9.2 *If all r(s,a) are non-negative and bounded and $Q^0(s,a) = 0$ then* $Q^{(n)} \rightarrow Q^*$

For the theorem to be true we assumed that we sample each (s,a) infinite number of times, thus we could look at each sequence separately.

How to assure that we visit all (s,a) ?

1. If we follow the greedy policy of $Q^{(n)}$ it may be not possible to check all pairs (s,a) even if we reach all states $s \in S$
2. We will follow from a state s the ε-greedy policy. With probability $1 - \varepsilon$ we choose the greedy action from s,
   and do a random policy from s with probability ε .
3. Choose the policy by:

   $\Pi^n(s,a) = \frac{exp - Q(s,a)/T}{\sum_b exp - Q(s,b)/T}$
   T determines if we do ε-greedy action (T $\rightarrow 0$) or random action (T $\rightarrow \infty$). We can look at this as if we part the time into sections of explorations(random) and of explotation(greedy).

On all 3 methods we can decide if we want to exploit the knowledge we have or get more information about the MDP

The next theorem is a weaker version of the main theorem.

Theorem 9.3 *Let* $r_{t+1} = (1 - \frac{1}{t})r_t + \frac{1}{t}(Hr_t + w_t)$
Assume that:

1. $E[w_t] = 0$ *and* $|w_t| \leq 1$
2. *H is a contracting mapping*
3. $|r_t| \leq D_0$

Then $r_t \rightarrow r^*$ *such that* $Hr^* = r^*$

Proof: Since H is contracting,
$||Hr - Hr^*|| \leq \beta ||r - r^*||$ for some $\beta < 1$.
Whithout loss of generality, assume that $r^* = 0$,
therefore $||Hr|| \leq \beta ||r||$.

Define: $D_{k+1} = (\beta + 2\varepsilon)D_k$ such that $\beta + 2\varepsilon < 1$.
We show by induction that exists a time t_k such that for every $t \geq t_k$ then $||r_t|| \leq D_k$

We bound r_t for $t > t_k$. since $||r_t|| \leq D_k$ then $||Hr_t|| \leq \beta D_k$ (the induction hypothesis).
We have: $r_{t+1} = \frac{t-1}{t} r_t + \frac{1}{t}(Hr_t + w_t) = \frac{t-l-1}{t} r_{t-l} + \frac{1}{t}\sum_{i=t-l}^{t} Hr_i + \frac{1}{t}\sum_{i=t-l}^{t} w_i$.
choose $l = t - t_k$ then : $||r_i|| \leq D_k$ and $||Hr_i|| \leq \beta D_k$.
Since $E[w_i] = 0$ and $|w_i| \leq 1$ we have that:$\frac{1}{t}\sum_{i=t-l}^{t} w_i \to 0$. Therefore $\frac{1}{t}\sum_{i=t-l}^{t} w_i \leq \varepsilon D_k$ for all but a finite number of t.

We continue: $r_t \leq \frac{t_k - 1}{t} r_{t_k} + \frac{1}{t}\sum_{i=t-l}^{t} Hr_i + \frac{1}{t}\sum_{i=t-l}^{t} w_i \leq \frac{t_k}{t} D_k + \frac{t-t_k}{t}\beta D_k + \frac{t-t_k}{t}\varepsilon$

For $t \geq t_k/\varepsilon$ we get:
$r_t \leq \varepsilon D_k + \beta D_k + \varepsilon D_k = (\beta + 2\varepsilon)D_k = D_{k+1}$

Therefore exists t_{k+1} such that $r_t \leq D_{k+1}$
Thus when $t \to \infty$ then $||r_t|| \to r^* = 0$. □