CS 282r: Reinforcement Learning

Instructor: Finale Doshi-Velez

Dustin Tran

Spring 2015

Contents

Preface

This document is a compilation of lecture notes for Professor Finale Doshi-Velez's CS 282r course in Harvard, during the term of Spring 2015. I do not claim these notes to be fully accurate or comprehensive, nor do I claim this document holds any officiality over the course. It contains exposition subject to my discretion. Please send feedback, corrections, etc. to dtran@g.harvard.edu. More broadly, I can be reached at dustintran.com.

CS 282r: Reinforcement Learning Lecture 01

Introduction to Reinforcement Learning

Lecturer: Finale Doshi-Velez *Date: January 29, 2015*

Motivation behind RL using an example of finding the optimal way to reach Harvard from Central Square (various routes of public transportation and walking), logistics of the course, a demo in class, brief description of Markov decision processes.

CS 282r: Reinforcement Learning Lecture 02

Markov decision processes (MDPs) and how to solve them

Lecturer: Finale Doshi-Velez *Date: February 3, 2015*

2.1 Markov decision processes (MDPs)

We follow Sutton & Barto, Ch. 3-4. Recall that a *Markov decision process* (MDP) depends on the collection of sets

$$\{S, A, R, T, \gamma\}, \tag{2.1}$$

where S is the states, A the actions, $R = R(s, a, s')$ the reward function, $T = T(s' \mid s, a)$, and γ the discount factor in $[0, 1]$. Note that T depends only on the action and the state of the most recent, not the entire history. It is also commonly used as $P(s' \mid s, a)$.

Figure 1

Goal: $\max \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$ using the *policy* $\pi(a|s)$. One can marginalize the action: $P(s'|s) = \sum_a \pi(a|s)T(s'|s, a)$, which is called a *Markov process*, or *Markov chain*.

Property 2.1 (Markov chains)**.** *Let $P_{ss'}$ be the transition matrix $P(s \to s')$, $(P_{ss'})^m$ be $P(s \to s'$ after m steps$)$.*

- *Can we get from any s to any s'? If yes, then the Markov chain is* ergodic*, or a* uni-chain*.*
- *Can we get "stuck"? If yes, then this is called the* absorbing state*.*

Definition 2.2. *$p(s)$ is a* stationary distribution *if where $p(s') = \sum_s p(s' \mid s)p(s)$, $p(s) = p(s')$.*

Figure 2. Stationary distribution is $(0, 0.5, 0.5)^T$.

Theorem 2.3. *The limiting distribution of a Markov chain where $m \to \infty$ is a stationary distribution.*

Note that $(P_{ss'})^m = AD^m A^T$, where D is some matrix with eigenvalues on the diagonal. The largest eigenvalue is always 1xx TODO: xx , and thus the second largest eigenvalue determines the convergence rate as it decays the slowest.

2.2 Given a policy: induced process value

Recall that each r_i leads to a new state s_{i+1}, and thus we are concerned with evaluating a policy. We define the *value function*

$$V^\pi(s) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r_t \mid \text{start in } s, \text{ do } \pi\Big] \tag{2.2}$$

and the *action-value function*

$$Q^\pi(s, a) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r_t \mid \text{start in } s, \text{ do } a, \text{ do } \pi\Big] \tag{2.3}$$

The value function estimates how *good it is* for the agent to be in a given state, and the action-value function estimates how good it is to perform a given action in a given state. Note that both are difficult to calculate as it sums over infinite time and there is the assumption that one knows the transition matrix.

$$V^\pi(s) = \sum_a \pi(a \mid s) \sum s' P(s' \mid s, a)[r(s, a, s') + \gamma \mathbb{E}[\sum \gamma^t r_t \text{ from } s'] \tag{2.4}$$

$$= \sum_a \pi(a \mid s) \sum s' P(s' \mid s, a)[r(s, a, s') + \gamma V^\pi(s')] \tag{2.5}$$

(**??**) is known as the *Bellman equation*. One can solve this by rearranging this into a linear system, or by using dynamic programming. We can use the iterative scheme

$$V_{k+1} = \sum_a \pi(a \mid s) \sum s' P(s' \mid s, a)[r(s, a, s') + \gamma V_k] \tag{2.6}$$

We can use this to model nonstationary distributions. One can also prove that the value update will converge to $V^\pi(s)$ using what is known as the *contraction property*:

(a) Define the Bellman operator, i.e., the update, as $TV \to V'$.

(b)

Claim 2.4. $\|TV - TV'\|_\infty \le \gamma \|V - V'\|$.

[Contraction property]

$$\|TV - TV'\|_\infty = \| \sum_a \pi(a \mid s) \sum s' P(s' \mid s, a)[r(s, a, s') + \gamma V(s')] - \sum_a \pi(a \mid s) \sum s' P(s' \mid s, a)[r(s, a, s') + \gamma V'(s')] \| \tag{2.7}$$

$$= \| \sum_a \pi(a \mid s) \sum s' P(s' \mid s, a)[\gamma(V(s') - V'(s'))] \| \tag{2.8}$$

$$\le \sum_a \pi(a \mid s) \sum s' P(s' \mid s, a) \gamma \|V(s') - V'(s')\|_\infty \tag{2.9}$$

$$= \gamma \|V(s') - V'(s')\|_\infty \tag{2.10}$$

$$\tag{2.11}$$

2.3 Improving/optimal policies

(a) One can compute V^π with the Bellman operator.

(b)

Theorem 2.5 (Policy improvement theorem)**.** *If $Q_\pi(s, \pi'(s)) \ge V_\pi(s)$, then $V_{\pi'}(s) \ge V_\pi(s)$.*

$$Q^\pi(s, a) = \sum s' P(s' \mid s, a)[r(s, a, s') + \gamma \mathbb{E}[\sum \gamma^t r_t \text{ from } s']$$

Choose π' that maximizes $Q^\pi(s, a)$. Can show that if $V_{\pi'}(s) = V_\pi(s)$, then π and π' are optimal.
To decide on the next policy, one can thus use the *Bellman operator criterion*:

$$V_{\pi'}(s) = \max_a \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma V_\pi(s)] = \max_a V_\pi(s')$$

Note that this is computationally tractable and furthermore can be found in polynomial time! This is better than the naive approach of combinatorially checking each branch of decisions individually.

(a) Policy iteration xx TODO: longer arrows for labels xx

$$\pi \to_{evaluation} V \to^{improve} \pi' \to \pi'' \to V'' \to \cdots$$

(b) Value iteration

$$V_{k+1}(s) \leftarrow \underbrace{\max_a}_{improve} \underbrace{\sum_{s'} p(s' \mid a, s)[r(s, a, s') + \gamma V_k(s')]}_{evaluation}$$

That is, it does improvement and evaluation in one step.

(c) Linear programming, which is used most in the old literature. Let $\mu(s) > 0$ for all s.

$$\min_{V(s)} \sum_s \mu(s) V(s) \text{ s.t. } V(s) \geq \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma V(s')]$$

In the end, all three are equivalent deterministic methods. Randomized approaches allow one to solve these reasonably efficiently with approximations.

Next time: Monte Carlo approaches (Sutton & Barto, Ch. 5, 6 Intro to TD)

- No reading response.
- There will be a concept check!

CS 282r: Reinforcement Learning Lecture 03

Monte Carlo approaches

Lecturer: Finale Doshi-Velez *Date: February 5, 2015*

Today:

(a) Demo: We see a program showing policy and value iteration in practice in order to solve paths to reach an endpoint in a maze.

(b) Practice concept check

(c) Monte Carlo approaches

(d) Real check

xx TODO: Exercises! xx

3.1 Monte Carlo approahces

Two cases: Policy evaluation (given π) and control (find optimal π)

3.1.1 Policy evaluation

(a) Generate a sequence $s_1 r_1 s_2 r_2 \cdots$

(b) What state do we want the value for? Denote it as state s^*. For example, say it's at s_3.

(c) Look at all rewards after 1st occurrence of s^*. The discounted return is

$$G = r_3 + \gamma r_4 + \gamma^2 r_5 + \cdots$$

(d) Do this m times. Then for sufficiently large simulations m,

$$V(s^*) = \sum_m G_m$$

This is an unbiased estimator, and note that the convergence does upon the variance of the estimate.

For policy improvement, we need $Q(s,a)$. If you set your simulator, "exploring starts" by obtaining a large number of $s^*a^* \to ... \to G$ is possible as it will be a consistent estimator but it is not practical.

Can we do better, i.e., without requiring setting the simulator? There are two approaches: *on-policy*, try to improve from what we have $\pi \to \pi'$, versus *off-policy*, try to evaluate/find π^*/π' given π.

On-policy: How do we always ensure that we see $Q(s,a)$ for all possible actions a? One approach is ϵ-greedy action selection. Suppose we have a run with an ϵ-greedy policy π. Then for $Q_\pi(s,a)$ estimated from m runs,

$$\begin{aligned} Q(s, \pi'(s)) &= \sum_a \pi'(a \mid s) Q_\pi(s,a) && (3.12)\\ &= \frac{\epsilon}{|A|} \sum_a Q_\pi(s,a) + (1-\epsilon) && (3.13)\\ &= \cdots && (3.14)\\ &\geq \sum_a \pi(s \mid a) Q_\pi(s,a) = V(s) && (3.15) \end{aligned}$$

Off-policy: Suppose we have an explore policy $\pi(a \mid s) > 0$ and we aim to evaluate some π'. Let S_T be the sequence $sarar \cdots$ for T. Then

$$p(S_T) = \prod_{t=1}^{T} \pi(a_t \mid s_t) p(s_{t+1} \mid s_{t,a}) \tag{3.16}$$

We thus aim to compute $\mathbb{E}_{\pi'}[\sum \gamma^t r_t]$. We can obtain draws from another policy π by applying importance sampling:

$$\mathbb{E}_q[f(x)] = \sum_x q(x) f(x) \tag{3.17}$$

$$= \sum_x \frac{q(x)}{p(x)} p(x) f(x) \tag{3.18}$$

$$= \mathbb{E}_p[\frac{q(x)}{p(x)} f(x)] \tag{3.19}$$

$$\approx \frac{1}{N} \sum_{x \sim p}^{N} \frac{q(x)}{p(x)} f(x) \tag{3.20}$$

Thus we have

$$1/N \sum^{N} \frac{\prod \pi'(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t)}{\prod \pi(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t)} G = 1/N \sum^{N} \frac{\prod \pi'(a_t \mid s_t)}{\prod \pi(a_t \mid s_t)} G \tag{3.21}$$

3.2 Concept check

xx TODO: xx

xx TODO: enumerate xx 1. Write down as MDP. What is the state space? 2. What might the form of the optimal policy look like? 3. Compute the value of the policy "always quit" for current points being 0, 1, 2, 3, 4, 5, 6. 4. Write down equation

xx TODO: transfer everything from paper xx

Next time: TD Methods (Sutton & Barto, Ch. 6, parts of 7)

CS 282r: Reinforcement Learning Lecture 04

Temporal difference (TD) methods

Lecturer: Finale Doshi-Velez *Date: February 12, 2015*

4.1 Practical 0 discussion

Recall that the main update in the SARSA algorithm is

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s,a) + \gamma Q(s',a') - Q(s,a)) \tag{4.22}$$

- The learning rate α has a huge effect and should not be a constant intuitively, as it should decay with the number of iterations, i.e., $\sum \alpha_i = \infty$, $\sum \alpha_i^2 < \infty$.
- Comparing rewards over episodes to cumulative rewards over iterations: The former displays convergence based on value where it ends and thus more easily determines convergence rate compared to other algorithms;; the latter displays which algorithm performs better in the long run.

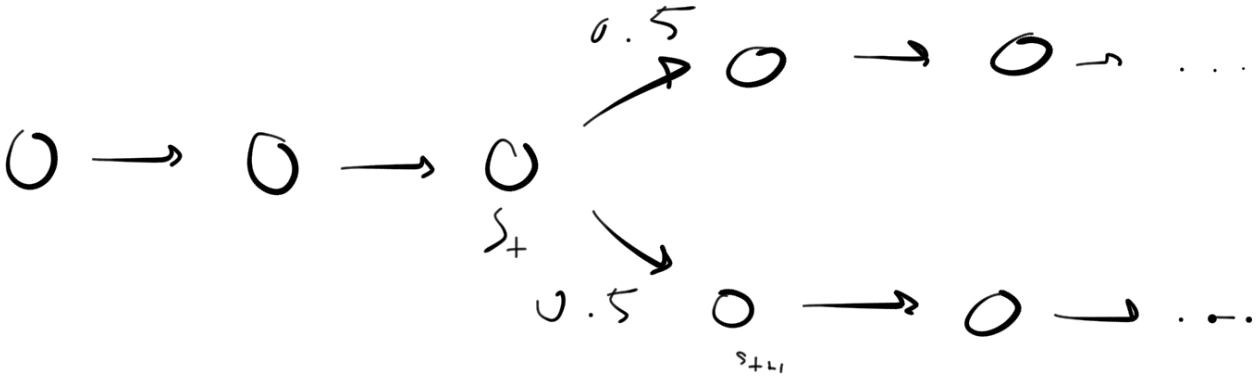

Figure 4.1: Example of an MDP.

4.2 TD methods

Recall that there is a value function and action-value function

$$V_\pi(s_t) = \underbrace{\mathbb{E}_\pi[\sum \gamma^t r_t \mid s = s_t]}_{\text{Monte Carlo simulation}} = \underbrace{\mathbb{E}_\pi[r_t + \gamma V_\pi(s_{t+1}) \mid s_t]}_{\text{value iteration/dynamic prog.}} \tag{4.23}$$

The advantage of Monte Carlo methods is that it does not require explicit information about the model, although value iteration is faster given such an assumption by using previously stored values via $V_\pi(s_{t+1} \mid s_t)$. TD methods are a compromise between both:

$$V(s) \leftarrow V(s) + \alpha(R + \gamma V(s') - V(s)) \tag{4.24}$$

It still uses some sample s', but it does not traverse the whole world in order to make a decision.

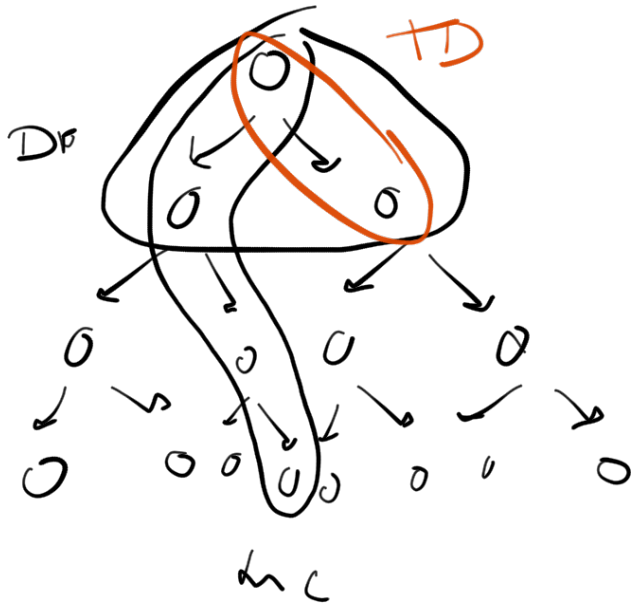

Figure 4.2: Comparison of TD, MC, and DP by the states each moves to in order to form an update.

Example 4.6. *There are two states A and B. At B, get $r = 0$ one time, $r = 1$ 6 times. At A we go to B and get 0. Then the value of B is $6/8 = 3/4$, and the value of A is either $3/4$ according to TD or 0 according to MC.*

Using an ϵ-greedy policy with respect to Q, there is an on-policy and off-policy learning procedure for TD. *SARSA*:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R + \gamma Q(s',a') - Q(s,a)); w \tag{4.25}$$

Q-learning:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R + \gamma \max_{a''} Q(s',a'') - Q(s,a)); w \tag{4.26}$$

4.3 Concept check

Let $\alpha = 1$ and $\gamma = 1$. Set the initial values V_0 as follows:

(a) What would one step of value iteration do? Write down the update equation and the V_1.

$$V_{k+1}(s) = \max_a \sum_{s'} p(s' \mid a, s)[r(s,a) + \gamma V_k(s')] \tag{4.27}$$

(b) What would MC with ABCF do? Write down the update equation and V_1.

(c) What would TD with ABCF do? Write down the update equation and V_1.

(d (test)) Consider a version of TD that uses two steps of experience instead of one.

$$V_{k+1}(s) = R(s) + \gamma R(s') + \gamma^2 V(s'') - V(s) \tag{4.28}$$

Interestingly, this extends to a $TD(\lambda)$ family of algorithms, where we classify TD error as

$$\underbrace{R(s_1) + \gamma R(s_2) + \gamma^2 R(s_3) + \cdots + \gamma^n V(s_{n+1})}_{G^{(n)}} - V(s) \tag{4.29}$$

The trade-off is to look more ahead but less bias. Moreover, one can show that

$$\|\mathbb{E}_\pi[G^(n)] - V_\pi(s)\|_\infty \leq \gamma^n \|V(s) - V_\pi(s)\|_\infty \tag{4.30}$$

For $\lambda \in [0, 1]$, the $TD(\lambda)$ update proceeds as follows:

$$V(s) \leftarrow V(s) + \alpha[(1-\lambda) \sum_{n=1}^{\infty} \lambda^n G_{(t)}^n \tag{4.31}$$

$\lambda = 1$ corresponds to MC and $\lambda = 0$ coresopnds to $TD(0)$.

Next time: E3 paper. Focus specifically on Section 5 and 5.1. There will be a concept check on it too.

CS 282r: Reinforcement Learning Lecture 05

E^3 algorithm

Lecturer: Finale Doshi-Velez *Date: February 17, 2015*

5.1 Discussion

- Create an algorithm for MDPs that has finite/polynomial resources
- Important idea - mixing time of a policy π
- Known and unknown states, with balanced wandering for exploration
- Assumption: know optimal policy value (valid?): if you always explore some, then you don't need this assumption

5.2 Main idea of E^3

The main influence in the paper aside from the polynomial time guarantee is to define states as *known* and *unknown*. Known states are constructed into a subset of the MDP as M_S, the known MDP, which has transitions between known that are the same as the MDP. Unknown simply does balanced wandering. Let $\widehat{M}_S$ be an approximation of the M_S. As we shall see later in the course, more sophisticated algorithms use the same concept, but generalized where states can be roughly known or unknown, and to what extent inbetween.

Figure 5.3: Induced Markov decision process M_S in the larger Markov decision process M.

Thus we solve two policies.

(a) Exploit policy, set $R_{unknown} = 0$. Will exploit in M_S.

(b) Explore policy, set $R_{unknown} = R_{max}$. Will try to go to unknown.

Notation used in paper:

1. MDPs: $P_M^a(i,j) = P_M(j \mid i,a)$, $0 < R_M(i) < R_{max}$.

2. Policies: map $\pi : \{1, \ldots, N\} \to \{a_1, \ldots, a_K\}$.

3. T-path: $\mathfrak{p} : i_1, i_2, \ldots, i_{T+1}$. Note that is fixed/given. Given a MDP M and a policy π, we can evaluate the probability of a T-path $P_M^\pi[\mathfrak{p}] = \prod_k^T P_M^{\pi(k)}(i_k i_{k+1})$.

4. return for MDP M is U_M, where

$$U_M(\mathfrak{p}) = \frac{1}{T}[R_i + \cdots + R_{i+T}] \tag{5.32}$$

$$U_M^\pi(i, T) = \sum_{\mathfrak{p}} P_M^\pi[\mathfrak{p}] U_M[p] \tag{5.33}$$

   The sum sums over all paths starting at i going T steps. Let U_M^K be the optimal value and U_M^π is the value for π as $T \to \infty$.

5. (Weak) *mixing time* T

$$\min T \text{ s.t. } |U_M^\pi(i, T) - U_M^\pi| < \epsilon \text{ for all } i$$

   This is "weaker" than the actual mixing time of MDP, as it only needs to mix for some subset of MDP that discovers the optimal policy.

Questions

1. Does a path in $\widehat{M}_S$ get similar rewards to a path in M_S? (Simulation lemma)

2. Can we exploit in M_S or quickly escape? (Exploit or explore lemma)

5.2.1 Simulation lemma

Definition 5.7. *$\widehat{M}$ is an α-approximation of the MDP M if for all i, $R_{\widehat{M}}(i) \in R_M(i) \pm \alpha$ and $P_{\widehat{M}}^\alpha(i, j) \in P_M^a(i, j) \pm \alpha$. Note that α is used in two places to mean two different values.*

Lemma 5.8 (Simulation lemma)**.** *How small does α have to be for $U_{\widehat{M}}^\pi(i, T)$ to be within $U_M(i, T) \pm \epsilon$ with probability $1 - \delta$?*

Proof. If within α, how does that translate into ϵ? Consider all paths $\mathfrak{p}$. Some will involve a β-small transition (unlikely paths), i.e., we will bound all paths with small probability by the value

$$\underbrace{\beta}_{pr(small)} \underbrace{N}_{\text{num. states}} \underbrace{\mathrm{T}}_{T \ times} \underbrace{G_{max}^T}_{\text{max value}}$$

This comes from $P(A \cup B) \leq P(A) + P(B)$, where we upper bound all such β-small transitions of T paths.

Other paths are not small: $P_{\widehat{M}}(i, j)$ is within $P_M(i, j) \pm \alpha$ implies $P_M(i, j)(1 + \Delta)$, where $\Delta = \alpha/\beta$. Then since $P_M^\pi[\mathfrak{p}] = \prod P_M^\pi(i, j)$, then $P_{\widehat{M}}^\pi[\mathfrak{p}]$ is within $(1 + \delta)^T P_M^\pi[\mathfrak{p}]$. Similarly, $U_{\widehat{M}}[\mathfrak{p}]$ is within $U_M[p] \pm \alpha$, which implies $U_{\widehat{M}}^\pi$ is within $(1 + \Delta)^T [U_\pi^M(i, T) - \alpha] - \epsilon/4$, where the first term regards big paths and second term regards small paths. Then we can use this to choose α such that we have error ϵ. □

5.2.2 Exploit or explore lemma

We first require the following property.

Property 5.9 (Hoeffding bound on Bernoulli variables)**.** *If $Z_1, \ldots, Z_n$ are i.i.d. Bernoulli r.v.s $\{0, 1\}$ with probability p, then*

$$P(|\widehat{p} - p| > \epsilon) \leq \exp(-2N\epsilon^2)$$

Let δ be the LHS. Then $N = 1/(2\epsilon^2)\ln(1/\delta)$

Consider the two state example, transitioning to two different states each with probabilities p and q respectively. Let $\widehat{p}$ be within $p \pm \epsilon$. $|\widehat{p} - \widehat{q}| = |p - q| + 2\epsilon$, by expanding $|(p + \epsilon) - (q - \epsilon)|$.

Thus there are two sources of error: δ_A, δ_B. If we want $\delta = \delta_A + \delta_B$, consider $\delta_A = \delta_B = \delta/2$, $\epsilon_A = \epsilon_B = \epsilon/2$.

Lemma 5.10 (Exploit or explore lemma)**.** *Can we exploit in M_S or quickly escape?*

Proof. Suppose there exists a situation

$$U^{\pi^*}_{M_S}(i, T) < U^{\pi^*}_{M}(i, T) - \alpha,$$

where π^* is optimal in M. See rest of proof in paper. □

5.3 Concept check

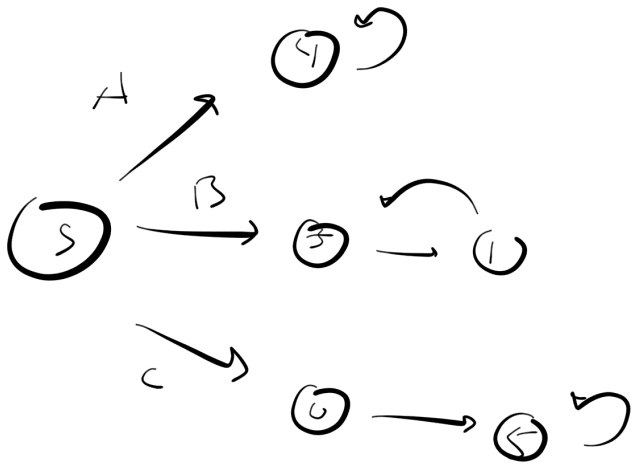

Figure 5.4: Markov decision process with starting point at state S with action space $\{A, B, C\}$. Rewards are denoted by the node label.

Let discount factor be denoted as γ.

Exercise 5.11. *What is the value of policies A, B, C?*

Solution.

- Path A: $V_\pi(s_t) = 4 + \gamma 4 + \gamma^2 4 + \cdots = 4\sum_{t=0}^{\infty}\gamma^t = 4/(1-\gamma)$.
- Path B: $V_\pi(s_t) = 5 + \gamma + \gamma^2 5 + \gamma^3 + \cdots = 5\sum_{\text{t even}}^{\infty}\gamma^t + \sum_{\text{t odd}}^{\infty}\gamma^t = 5\sum_{\text{t even}}^{\infty}\gamma^t + \gamma\sum_{\text{t even}}^{\infty}\gamma^t = 5/(1-\gamma^2) + \gamma/(1-\gamma^2)$
- Path C: $V_\pi(s_t) = 0 + 5\gamma/(1-\gamma^2)$.

□

Exercise 5.12. *When is it better to do B instead of A?*

Solution. Recall $\sum_{t=0}^{\infty}\gamma^t = 1/(1-\gamma)$.

To see when it's better to do B, compare γ when

$$4/(1-\gamma) \le 5/(1-\gamma^2) + \gamma/(1-\gamma^2)$$

For $\gamma > 4/5$, prefer C, and for $\gamma < 1/3$, prefer B. □

Exercise 5.13 (Concept check)**.** *Undiscounted case with fixed time T. When is A, B, or C best?*

Solution. A is best for 2 to 4, B for 1, C for 5 onwards □

CS 282r: Reinforcement Learning Lecture 06

R-MAX, Thompson sampling

Lecturer: Finale Doshi-Velez *Date: February 19, 2015*

6.1 A brief overview of the course

- Introduced RL framework, composed of an agent acting on the world and receiving states and rewards; this is formalized by MDPs.
- One way of solving them is by using the Bellman equation.
- Model-free: SARSA, Q-learning (TD methods) in order to directly compute
- Model-based: E^3, R-MAX, Thompson sampling in order to build an approximate model and solve

Next we shall see more model-based methods which are more intelligent about the in-between of not being completely known or unknown. Then extensions: how to set parameters; large state spaces; batch methods

6.2 R-MAX

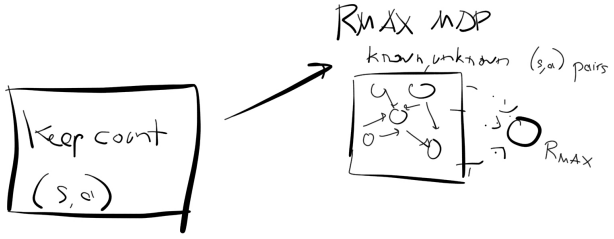

Figure 6.5: R-MAX

Why does this work? (Simulation lemma) Known MDPs are approximately the true one for returns; we will explore quickly, i.e., $P(\text{go to unknown}) > \epsilon$. Note that unlike E^3, R-MAX does not necessarily explore all spaces, as a significant enough discount factor can cause it to stay with the short term.

6.3 Thompson sampling

All previous model-based approaches we've encountered have been frequentist, i.e., there is a belief of a true parameter specifying the MDP and we reduce the bounds of our confidence interval. Thompson sampling is an example of a Bayesian method, which specifies a prior on one's belief about what the MDP is most like, which updates it with data and obtains a posterior.

Assume rewards are deterministic (for simplicity only; it may be generalized). Specify a prior over transitions.

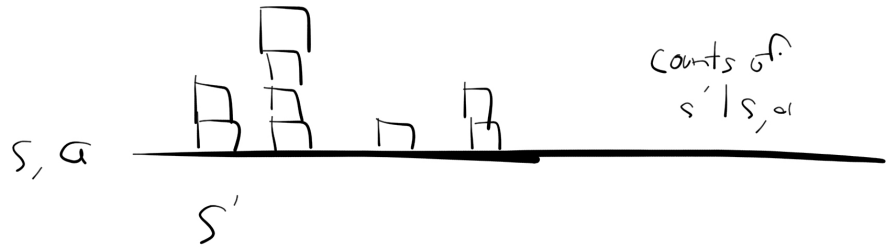

Figure 6.6: Treating transitions $P(s' \mid s, a)$ as count proportions represented under a multinomial distribution.

Hence we can represent transitions for a state-action pair as multinomial, and apply a Dirichlet distribution, in which a sample from a Dirichlet represents sampling a single MDP.

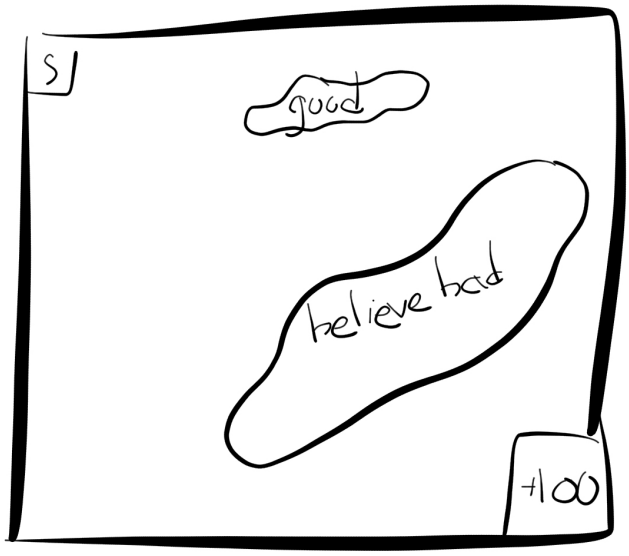

Figure 6.7: Example with strongly misspecified prior distribution.

Asymptotically, the prior does not matter but for finite N or a strong discount factor γ, it is very hard to overcome the biases. Note that for high pseudocounts, say, one million on each category of α_k, then we are strongly sure that all transition probabilities are uniform. If the prior is weak, then it doesn't as strongly affect the posterior.

6.4 Concept check: Shaping rewards

Following Ng and Russel, "Shaping Rewards."

Exercise 6.14. *Suppose I have 2 MDPs, $Q^*_M(s,a)$ and $Q^*_{M'}(s,a) = Q^*_M(s,a) + f(s)$. Is the optimal policy for M' the same or different than M?*

Answer. Consider $Q(s,\cdot)$, which depends on $\arg\max_a Q(s,a)$. Hence any change by $f(s)$ doesn't matter. □

Exercise 6.15. *Suppose I change each reward*

$$R'(s,a,s') = R(s,a,s') + \Phi(s) - \gamma\Phi(s') \tag{6.34}$$

Does the optimal policy change? Hint: start with Bellman equation for $Q^*_M(s,a)$ and relate it to $Q^*_{M'}(s,a)$

Answer. By the Bellman equation,

$$\begin{aligned} Q^*_M(s,a) &= \mathbb{E}_{s'}[R(s,a,s') + \gamma \max_a Q^*_M(s',a')] && (6.35)\\ &= \mathbb{E}_{s'}[R(s,a,s') + \Phi(s) - \Phi(s) + \gamma\Phi(s') - \gamma\Phi(s')\gamma \max_a Q^*_M(s',a')] && (6.36)\\ &= \mathbb{E}_{s'}[R'(s,a,s') - \Phi(s) + \gamma \max_a [Q^*_M(s',a') + \Phi(s')]] && (6.37) \end{aligned}$$

Translate by $\Phi(s)$ does not matter since it is in expectation of s', and similarly, the maximum action is not concerned with $\Phi(s')$. □

CS 282r: Reinforcement Learning Lecture 07

BOSS, MBIE

Lecturer: Finale Doshi-Velez *Date: February 24, 2015*

Main idea:

- Constructing optimistic transition functions

Recall that in the RL framework, we obtain history $sarsara\cdots$ and in our approach, we construct an approximate model, solve, etc.

Theorem 7.16 (PAC-MDP)**.** *For an algorithm to be PAC-MDP:* $\text{poly}(|S|, |A|, 1/\epsilon, 1/\delta, 1/(1-\gamma)$ *or* $T)$ *samples is required to perform near-optimally.*

There are multiple conditions to do this:

(a) exploration: optimistic value function

(b) simulation lemma: accuracy for known states, i.e., exploit: if known, we can do well

(c) exploration ends: visits to unknown states $< |S|A|B$, where $B \in \mathbb{R}$ is a constant

There's been little development on the simulation lemma, as most invoke Hoeffding's inequality, c.f., E^3 and R-MAX.

7.1 MBIE (Model-based Interval Exploration)

For great optimistic value functions, we set optimistic transitions.

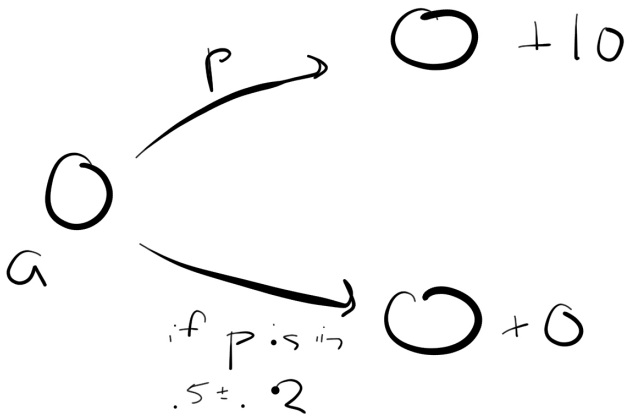

Figure 7.8: MDP starting at left, with action a corresponding to Bernoulli(p) arrive to state with reward 0 if p is in 0.5 ± 0.2, reward $+10$ otherwise. Let's believe $\widetilde{p} = 0.7$ and see what happens.

More formally, the error in the transition

$$T(\cdot \mid s, a) \leq d\sqrt{\frac{k(s,a)}{n(s,a)}}, \tag{7.38}$$

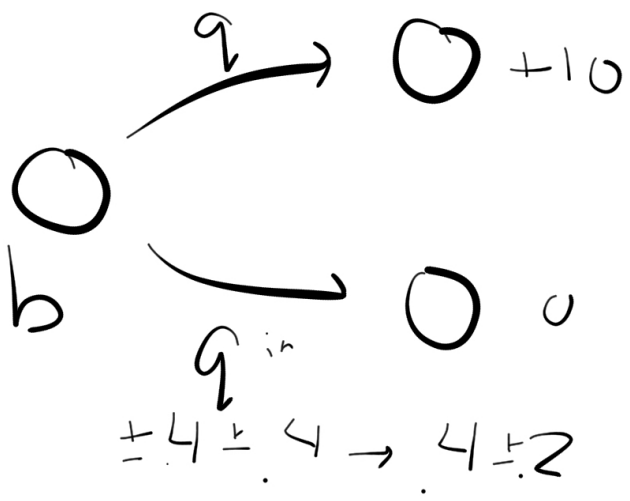

Figure 7.9: Using Bernoulli(q), obtain reward 0 if q in 0.4 ± 0.4. Let's say $\widetilde{q} = 0.8$.

where $k(\cdot, \cdot)$ is the number of states you can go to after s and do a and $n(\cdot, \cdot)$ is the number of visits to s, a.

$$Q(s,a) = \widehat{R}(s,a) + \max_{\widetilde{T} \in \text{C.I.}} \gamma \sum_{s'} \widetilde{T}(s' \mid s,a) \max_{a'} Q(s',a') \tag{7.39}$$

See figures in paper of its comparison to R-MAX.

7.2 BOSS (Best of Sampled Set)

1. Keep a posterior over MDPs.
2. Sample B MDPs from the posterior, where each has actions $a_1, \ldots, a_k$.
3. Create optimistic MDP $M\#$, where $M\#$ has the same states $\{s_1, \ldots, s_N\}$ and combined actions $\{a_{ij}\}_{i=1,\ldots,B, j=1,\ldots,k}$.
4. Choose an action based on the best MDP for that current state action pair.

7.3 Concept check

Reduced-order models for data-limited RL (Joseph & Roy).

Exercise 7.17. *Write down value of action A B?*

Answer. Once can simply compute the expectation of each state and action pair:

$$V^A(s) = 50(1-p_1) - 10p_1 \tag{7.40}$$
$$V^B(s) = 5(1-p_2) \tag{7.41}$$

□

Exercise 7.18. *Suppose that $p_1 = 1, p_2 = 1/5$. Compute values from part A. What is the better policy?*

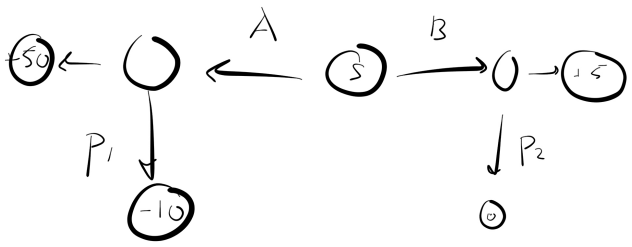

Figure 7.10: $P(fail) = p_1$, which sends us to -10, otherwise +50. $P(fail) = p_2$ in B. Let $\gamma = 1$.

Answer. $V^A(s) = -10$ and $V^B(s) = 4$. □

Exercise 7.19. *Suppose we think $p_1 = p_2$, and we've seen A 1 try, 1 fail and B 4 tries, 4 success. What is $\widehat{p}_{MLE}$ What is the optimal policy given $\widehat{p}_{MLE}$?*

Answer. Since $p_1 = p_2$, then $\widehat{p}_{MLE} = 1/5$. Then the optimal policy given it is A since $50(1-1/5)-10(1/5) > 5(1-1/5)$. □

Exercise 7.20. *Is there a value of $\widehat{p}$ in the misspecified model that still provides the optimal policy (choose B)? If not, explain why. If so, give an example.*

Answer. By direct calculation, one can show that in order to guarantee that the optimal policy of B is greater than A, given $\widehat{p}$, then $\widehat{p} > 45/55$. So yes. □

CS 282r: Reinforcement Learning Lecture 08

KWIK

Lecturer: Finale Doshi-Velez *Date: February 26, 2015*

8.1 Logistics

For Thompson sampling, only consider uncertainty in transitions. Practical 1 will be out of 13 points; 8 points relating to results and 8 relating to discussion.

Final projects

- More parameter exploration
  - Papers on optimal parameters
  - Bayesian optimization of parameters
- Choose a domain and try to solve it
  - Othello, Tetris, AI
  - Other large/real domain
- Theoretical
  - Extensions of stuff from class (transfer learning)
  - Literature review with in-depth discussion

MDP extensions: function approximations/larger MDPs, batch methods.

8.2 Knows What It Knows (KWIK)

Refer to the diagram comparing PAC, MB, and KWIK in the paper.

Example 8.21 (Conjunction learning)**.** *Input is a sequence of binary values* 001001100*. Output: true or false. Hypothesis class: if a certain conjunction is present, predict true, e.g., if* d_1 *and* d_2 *and* d_5*, predict true.*

Proof. MB: guess false always. If false, then you're okay. If true. Know any bit in string that's 0 cannot be part of conjunctions. Rule out all possibilities for each dim. Hence linear complexity.

KWIK: Because you can't make any mistakes, the adversary can keep providing you a bunch of False's, from which you will learn very little about. For a string of length n, there are 2^{n-1} possible scenarios where you cannot learn anything. Hence exponential complexity. □

8.2.1 KWIK for MDPs

Algs: If hypothesis class or inpute class are finite of size S, KWIK learns in S. (trivial)

Coin tossing to learn p(heads). With noisy labels $\{0, 1\}$, one can apply Hoeffding bounds.

Taking unions of KWIK learners on disjoin input spaces:

$$B(\epsilon, \delta) = \sum_i^k B_i(\epsilon, \delta/k) \tag{8.42}$$

Say you have K coints, adversary picks the coin, you have to predict p(heads).
MDP case:

1. Any MDP with uncertain transitions and rewards, but where there exist finite set of possible rewards $\{r_L, r_M, r_H\}$. Can map to MDP' with just uncertain transitions. To do this, simply augment your MDP by a factor of 3, i.e., for each possible state have $(s_i, r_L), (s_i, r_M), (s_i, r_H)$.

2. Learning an uncertain transition, which corresponds to learning a bunch of coin flips. $(s, a, s') \to p$.

8.3 Concept check

Related to practical 1.

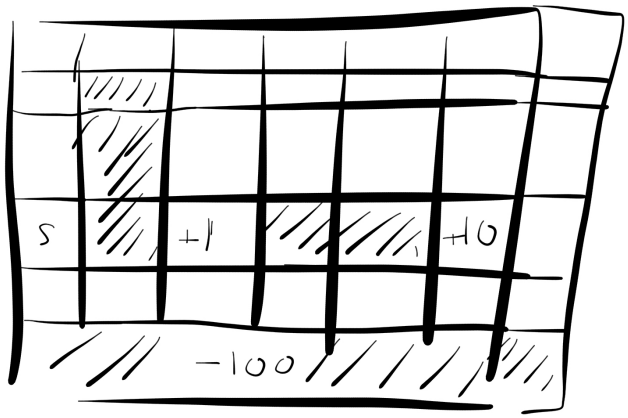

Figure 8.11: Cliffworld example. Walls are the blacked out boxes with no numeric annotated in them.

Let α be your transition noise, i.e., with probability α, environment does something random instead of a. Let ϵ be ϵ-greedy parameter.

Exercise 8.22. *(a) Let $\epsilon = 0$. Optimal policy if $\gamma = 0.1, \alpha = 0.5$?*

(b) $\gamma = 0.1, \alpha = 0.5$?

(c) $\gamma = 0.1, \alpha = 0$?

(d) $\gamma = 1, \alpha = 0.5$?

(e) $\gamma = 1, \alpha = 0$?

Concept check $\gamma = 1, \alpha = 0$. Now our policy explores with $\epsilon = 0.5$. What path will SARSA learn? What path will Q-learning learn?

Answer. $\gamma = 1$ is undiscounted, and so the optimal policy is to aim for the +10 and cycle. If $\gamma = 0.1$, then the optimal policy is to aim for the +1. The path to reach one of these two depends on α. If $\alpha = 0$, then there's no chance of falling off the cliff so it goes down and around, as it achieves the reward faster. If $\alpha = 0.5$, then there is a chance, so it will go up and around.

On the concept check: See Cliffworld example in Sutton & Barto (2012). SARSA is safer and will go around. Q-learning will not. □