Lecture 13: Fast Reinforcement Learning [1]

Emma Brunskill

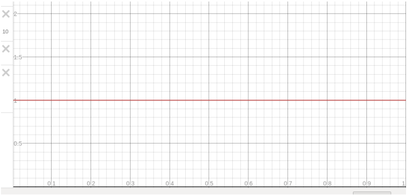
CS234 Reinforcement Learning

Winter 2020

[1] With a few slides derived from David Silver

Refresh Your Knowledge Fast RL Part II

- The prior over arm 1 is Beta(1,2) (left) and arm 2 is a Beta(1,1) (right figure). Select all that are true.
  1. Sample 3 params: 0.1,0.5,0.3. These are more likely to come from the Beta(1,2) distribution than Beta(1,1).
  2. Sample 3 params: 0.2,0.5,0.8. These are more likely to come from the Beta(1,1) distribution than Beta(1,2).
  3. It is impossible that the true Bernoulli parame is 0 if the prior is Beta(1,1).
  4. Not sure
- The prior over arm 1 is Beta(1,2) (left) and arm 2 is a Beta(1,1) (right). The true parameters are arm 1 $\theta_1 = 0.4$ & arm 2 $\theta_2 = 0.6$. Thompson sampling = TS
  1. TS could sample $\theta = 0.5$ (arm 1) and $\theta = 0.55$ (arm 2).
  2. For the sampled thetas (0.5,0.55) TS is optimistic with respect to the true arm parameters for all arms.
  3. For the sampled thetas (0.5,0.55) TS will choose the true optimal arm for this round.
  4. Not sure



Class Structure

- Last time: Fast Learning (Bayesian bandits to MDPs)
- **This time: Fast Learning III (MDPs)**
- Next time: Batch RL

Settings, Frameworks & Approaches

- Over these 3 lectures will consider 2 settings, multiple frameworks, and approaches
- Settings: Bandits (single decisions), MDPs
- Frameworks: evaluation criteria for formally assessing the quality of a RL algorithm. So far seen empirical evaluations, asymptotic convergence, regret, probably approximately correct
- Approaches: Classes of algorithms for achieving particular evaluation criteria in a certain set. So far for exploration seen: greedy, $\epsilon-$greedy, optimism, Thompson sampling, for multi-armed bandits

Table of Contents

Fast RL in Markov Decision Processes

- Very similar set of frameworks and approaches are relevant for fast learning in reinforcement learning
- Frameworks
  - Regret
  - Bayesian regret
  - Probably approximately correct (PAC)
- Approaches
  - Optimism under uncertainty
  - Probability matching / Thompson sampling
- Framework: Probably approximately correct

Fast RL in Markov Decision Processes

- Montezuma's revenge
- https://www.youtube.com/watch?v=ToSe_CUG0F4

Model-Based Interval Estimation with Exploration Bonus (MBIE-EB)

(Strehl and Littman, J of Computer & Sciences 2008)

Given ϵ, δ, m
$\beta = \frac{1}{1-\gamma}\sqrt{0.5\ln(2|S||A|m/\delta)}$
$n_{sas}(s, a, s') = 0$, $\forall s \in S$, $a \in A$, $s' \in S$
$rc(s, a) = 0$, $n_{sa}(s, a) = 0$, $\tilde{Q}(s, a) = 1/(1-\gamma)$, $\forall\ s \in S$, $a \in A$
$t = 0$, $s_t = s_{init}$
loop
    $a_t = \arg\max_{a \in \mathcal{A}} \tilde{Q}(s_t, a)$
    Observe reward r_t and state s_{t+1}
    $n_{sa}(s_t, a_t) = n_{sa}(s_t, a_t) + 1$, $n_{sas}(s_t, a_t, s_{t+1}) = n_{sas}(s_t, a_t, s_{t+1}) + 1$
    $rc(s_t, a_t) = \frac{rc(s_t, a_t)(n_{sa}(s_t, a_t) - 1) + r_t}{n_{sa}(s_t, a_t)}$
    $\hat{R}(s_t, a_t) = rc(s_t, a_t)$ and $\hat{T}(s'|s_t, a_t) = \frac{n_{sas}(s_t, a_t, s')}{n_{sa}(s_t, a_t)}$, $\forall s' \in S$
    while not converged **do**
        $\tilde{Q}(s, a) = \hat{R}(s, a) + \gamma \sum_{s'} \hat{T}(s'|s, a) \max_{a'} \tilde{Q}(s', a) + \frac{\beta}{\sqrt{n_{sa}(s, a)}}$, $\forall\ s \in S$, $a \in A$
    end while
end loop

Framework: PAC for MDPs

- For a given ϵ and δ, A RL algorithm $\mathcal{A}$ is PAC if on all but N steps, the action selected by algorithm $\mathcal{A}$ on time step t, a_t, is ϵ-close to the optimal action, where N is a polynomial function of $(|S|, |A|, \gamma, \epsilon, \delta)$
- Is this true for all algorithms?

MBIE-EB is a PAC RL Algorithm

Theorem 2. *Suppose that* ϵ *and* δ *are two real numbers between* 0 *and* 1 *and* $M = \langle S, A, T, \mathcal{R}, \gamma \rangle$ *is any MDP. There exists an input* $m = m(\frac{1}{\epsilon}, \frac{1}{\delta})$, *satisfying* $m(\frac{1}{\epsilon}, \frac{1}{\delta}) = O(\frac{|S|}{\epsilon^2(1-\gamma)^4} + \frac{1}{\epsilon^2(1-\gamma)^4} \ln \frac{|S||A|}{\epsilon(1-\gamma)\delta})$, *and* $\beta = (1/(1-\gamma))\sqrt{\ln(2|S||A|m/\delta)/2}$ *such that if MBIE-EB is executed on MDP* M, *then the following holds. Let* $\mathcal{A}_t$ *denote MBIE-EB's policy at time* t *and* s_t *denote the state at time* t. *With probability at least* $1-\delta$, $V_M^{\mathcal{A}_t}(s_t) \geqslant V_M^*(s_t) - \epsilon$ *is true for all but* $O(\frac{|S||A|}{\epsilon^3(1-\gamma)^6}(|S| + \ln \frac{|S||A|}{\epsilon(1-\gamma)\delta}) \ln \frac{1}{\delta} \ln \frac{1}{\epsilon(1-\gamma)})$ *timesteps* t.

A Sufficient Set of Conditions to Make a RL Algorithm PAC

- Strehl, A. L., Li, L., & Littman, M. L. (2006). Incremental model-based learners with formal learning-time guarantees. In Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence (pp. 485-493)

A Sufficient Set of Conditions to Make a RL Algorithm PAC

How Does MBIE-EB Fulfill these Conditions?

Table of Contents

Refresher: Bayesian Bandits

- **Bayesian bandits** exploit prior knowledge of rewards, $p[\mathcal{R}]$
- They compute posterior distribution of rewards $p[\mathcal{R} \mid h_t]$, where $h_t = (a_1, r_1, \ldots, a_{t-1}, r_{t-1})$
- Use posterior to guide exploration
  - Upper confidence bounds (Bayesian UCB)
  - Probability matching (Thompson Sampling)
- Better performance if prior knowledge is accurate

Refresher: Bernoulli Bandits

- Consider a bandit problem where the reward of an arm is a binary outcome $\{0, 1\}$ sampled from a Bernoulli with parameter θ
  - E.g. Advertisement click through rate, patient treatment succeeds/fails, ...
- The Beta distribution $Beta(\alpha, \beta)$ is conjugate for the Bernoulli distribution

$$p(\theta|\alpha, \beta) = \theta^{\alpha-1}(1-\theta)^{\beta-1}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}$$

  where $\Gamma(x)$ is the Gamma function.
- Assume the prior over θ is a $Beta(\alpha, \beta)$ as above
- Then after observed a reward $r \in \{0, 1\}$ then updated posterior over θ is $Beta(r + \alpha, 1 - r + \beta)$

Thompson Sampling for Bandits

Initialize prior over each arm a, $p(\mathcal{R}_a)$
loop
  For each arm a **sample** a reward distribution $\mathcal{R}_a$ from posterior
  Compute action-value function $Q(a) = \mathbb{E}[\mathcal{R}_a]$
  $a_t = \arg\max_{a \in \mathcal{A}} Q(a)$
  Observe reward r
  Update posterior $p(\mathcal{R}_a | r)$ using Bayes law
end loop

Bayesian Model-Based RL

- Maintain posterior distribution over **MDP** models
- Estimate both transition and rewards, $p[\mathcal{P}, \mathcal{R} \mid h_t]$, where $h_t = (s_1, a_1, r_1, \ldots, s_t)$ is the history
- Use posterior to guide exploration
  - Upper confidence bounds (Bayesian UCB)
  - Probability matching (Thompson sampling)

Thompson Sampling: Model-Based RL

- Thompson sampling implements probability matching

$$\begin{aligned}\pi(s, a \mid h_t) &= \mathbb{P}[Q(s,a) \geq Q(s,a'), \forall a' \neq a \mid h_t] \\ &= \mathbb{E}_{\mathcal{P},\mathcal{R}|h_t}\left[\mathbb{1}(a = \arg\max_{a \in \mathcal{A}} Q(s,a))\right]\end{aligned}$$

- Use Bayes law to compute posterior distribution $p[\mathcal{P}, \mathcal{R} \mid h_t]$
- **Sample** an MDP $\mathcal{P}, \mathcal{R}$ from posterior
- Solve MDP using favorite planning algorithm to get $Q^*(s, a)$
- Select optimal action for sample MDP, $a_t = \arg\max_{a \in \mathcal{A}} Q^*(s_t, a)$

Thompson Sampling for MDPs

Initialize prior over the dynamics and reward models for each (s, a), $p(\mathcal{R}_{as})$, $p(\mathcal{T}(s'|s, a))$
Initialize state s_0
loop
  Sample a MDP $\mathcal{M}$: for each (s, a) pair, sample a dynamics model $\mathcal{T}(s'|s, a)$ and reward model $\mathcal{R}(s, a)$
  Compute $Q^*_{\mathcal{M}}$, optimal value for MDP $\mathcal{M}$
  $a_t = \arg\max_{a \in \mathcal{A}} Q^*_{\mathcal{M}}(s_t, a)$
  Observe reward r_t and next state s_{t+1}
  Update posterior $p(\mathcal{R}_{a_t s_t}|r_t)$, $p(\mathcal{T}(s'|s_t, a_t)|s_{t+1})$ using Bayes rule
  $t = t + 1$
end loop

Check Your Understanding: Fast RL III

- Strategic exploration in MDPs (select all):
  1. Doesn't really matter because the distribution of data is independent of the policy followed
  2. Can involve using optimism with respect to both the possible dynamics and reward models in order to compute an optimistic Q function
  3. Is known as PAC if the number of time steps on which a less than near optimal decision is made is guaranteed to be less than an exponential function of the problem domain parameters (state space cardinality, etc).
  4. Not sure
- In Thompson sampling for MDPs:
  1. TS samples the reward model parameters and could use the empirical average for the dynamics model parameters and obtain the same performance
  2. Must perform MDP planning everytime the posterior is updated
  3. Has the same computational cost each step as Q-learning
  4. Not sure

Resampling in Coordinated Exploration

- Concurrent PAC RL. Guo and Brunskill. AAAI 2015
- Coordinated Exploration in Concurrent Reinforcement Learning. Dimakopoulou and Van Roy. ICML 2018
- https://www.youtube.com/watch?v=xjGK-wm0PkI&feature=youtu.be

Table of Contents

Generalization and Strategic Exploration

- Active area of ongoing research: combine generalization & strategic exploration
- Many approaches are grounded by principles outlined here
  - Optimism under uncertainty
  - Thompson sampling

Generalization and Optimism

- Recall MBIE-EB algorithm for finite state and action domains
- What needs to be modified for continuous / extremely large state and/or action spaces?

Model-Based Interval Estimation with Exploration Bonus (MBIE-EB)

(Strehl and Littman, J of Computer & Sciences 2008)

Given ϵ, δ, m
$\beta = \frac{1}{1-\gamma}\sqrt{0.5 \ln(2|S||A|m/\delta)}$
$n_{sas}(s, a, s') = 0$, $\forall s \in S$, $a \in A$, $s' \in S$
$rc(s, a) = 0$, $n_{sa}(s, a) = 0$, $\tilde{Q}(s, a) = 1/(1-\gamma)$, $\forall\ s \in S$, $a \in A$
$t = 0$, $s_t = s_{init}$
loop
    $a_t = \arg\max_{a \in \mathcal{A}} \tilde{Q}(s_t, a)$
    Observe reward r_t and state s_{t+1}
    $n_{sa}(s_t, a_t) = n_{sa}(s_t, a_t) + 1$, $n_{sas}(s_t, a_t, s_{t+1}) = n_{sas}(s_t, a_t, s_{t+1}) + 1$
    $rc(s_t, a_t) = \frac{rc(s_t, a_t)(n_{sa}(s_t, a_t) - 1) + r_t}{n_{sa}(s_t, a_t)}$
    $\hat{R}(s_t, a_t) = rc(s_t, a_t)$ and $\hat{T}(s'|s_t, a_t) = \frac{n_{sas}(s_t, a_t, s')}{n_{sa}(s_t, a_t)}$, $\forall s' \in S$
    while not converged **do**
        $\tilde{Q}(s, a) = \hat{R}(s, a) + \gamma \sum_{s'} \hat{T}(s'|s, a) \max_{a'} \tilde{Q}(s', a) + \frac{\beta}{\sqrt{n_{sa}(s, a)}}$, $\forall\ s \in S$, $a \in A$
    end while
end loop

Generalization and Optimism

- Recall MBIE-EB algorithm for finite state and action domains
- What needs to be modified for continuous / extremely large state and/or action spaces?
- Estimating uncertainty
  - Counts of (s,a) and (s,a,s') tuples are not useful if we expect only to encounter any state once
- Computing a policy
  - Model-based planning will fail
- So far, model-free approaches have generally had more success than model-based approaches for extremely large domains
  - Building good transition models to predict pixels is challenging

Recall: Value Function Approximation with Control

- For Q-learning use a TD target $r + \gamma \max_{a'} \hat{Q}(s', a'; \boldsymbol{w})$ which leverages the max of the current function approximation value

$$\Delta \boldsymbol{w} = \alpha(r(s) + \gamma \max_{a'} \hat{Q}(s', a'; \boldsymbol{w}) - \hat{Q}(s, a; \boldsymbol{w}))\nabla_{\boldsymbol{w}} \hat{Q}(s, a; \boldsymbol{w})$$

Recall: Value Function Approximation with Control

- For Q-learning use a TD target $r + \gamma \max_{a'} \hat{Q}(s', a'; \boldsymbol{w})$ which leverages the max of the current function approximation value

$$\Delta \boldsymbol{w} = \alpha(r(s) + r_{bonus}(s,a) + \gamma \max_{a'} \hat{Q}(s', a'; \boldsymbol{w}) - \hat{Q}(s, a; \boldsymbol{w})) \nabla_{\boldsymbol{w}} \hat{Q}(s, a; \boldsymbol{w})$$

Recall: Value Function Approximation with Control

- For Q-learning use a TD target $r + \gamma \max_{a'} \hat{Q}(s', a'; \boldsymbol{w})$ which leverages the max of the current function approximation value

$$\Delta \boldsymbol{w} = \alpha(r(s) + r_{bonus}(s,a) + \gamma \max_{a'} \hat{Q}(s', a'; \boldsymbol{w}) - \hat{Q}(s, a; \boldsymbol{w})) \nabla_{\boldsymbol{w}} \hat{Q}(s, a; \boldsymbol{w})$$

- $r_{bonus}(s, a)$ should reflect uncertainty about future reward from (s, a)
- Approaches for deep RL that make an estimate of visits / density of visits include: Bellemare et al. NIPS 2016; Ostrovski et al. ICML 2017; Tang et al. NIPS 2017
- Note: bonus terms are computed at time of visit. During episodic replay can become outdated.

Benefits of Strategic Exploration: Montezuma's revenge

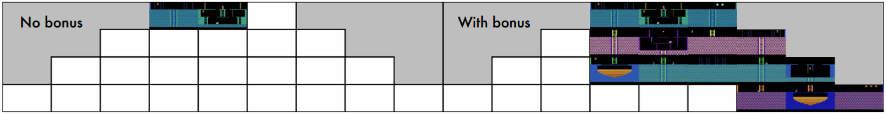


Figure 3: "Known world" of a DQN agent trained for 50 million frames with (**right**) and without (**left**) count-based exploration bonuses, in MONTEZUMA'S REVENGE.

Figure: Bellemare et al. "Unifying Count-Based Exploration and Intrinsic Motivation"

- Enormously better than standard DQN with ϵ-greedy approach

Generalization and Strategic Exploration: Thompson Sampling

- Leveraging Bayesian perspective has also inspired some approaches
- One approach: Thompson sampling over representation & parameters (Mandel, Liu, Brunskill, Popovic IJCAI 2016)

Generalization and Strategic Exploration: Thompson Sampling

- Leveraging Bayesian perspective has also inspired some approaches
- One approach: Thompson sampling over representation & parameters (Mandel, Liu, Brunskill, Popovic IJCAI 2016)
- For scaling up to very large domains, again useful to consider model-free approaches
- Non-trivial: would like to be able to sample from a posterior over possible Q^*
- Bootstrapped DQN (Osband et al. NIPS 2016)
  - Train C DQN agents using bootstrapped samples
  - When acting, choose action with highest Q value over any of the C agents
  - Some performance gain, not as effective as reward bonus approaches

Generalization and Strategic Exploration: Thompson Sampling

- Leveraging Bayesian perspective has also inspired some approaches
- One approach: Thompson sampling over representation & parameters (Mandel, Liu, Brunskill, Popovic IJCAI 2016)
- For scaling up to very large domains, again useful to consider model-free approaches
- Non-trivial: would like to be able to sample from a posterior over possible Q^*
- Bootstrapped DQN (Osband et al. NIPS 2016)
- Efficient Exploration through Bayesian Deep Q-Networks (Azizzadenesheli, Anandkumar, NeurIPS workshop 2017)
  - Use deep neural network
  - On last layer use Bayesian linear regression
  - Be optimistic with respect to the resulting posterior
  - Very simple, empirically much better than just doing linear regression on last layer or bootstrapped DQN, not as good as reward bonuses in some cases

Table of Contents

Summary: What You Are Expected to Know

- Define the tension of exploration and exploitation in RL and why this does not arise in supervised or unsupervised learning
- Be able to define and compare different criteria for "good" performance (empirical, convergence, asymptotic, regret, PAC)
- Be able to map algorithms discussed in detail in class to the performance criteria they satisfy

Class Structure

- Last time: Fast Learning (Bayesian bandits to MDPs)
- **This time: Fast Learning III (MDPs)**
- Next time: Batch RL