Last time: Fast Learning (Bayesian bandits to MDPs)

This time: Fast Learning III (MDPs)

Next time: Meta-learning (Guest speaker: Chelsea Finn)
Over next couple 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

Approaches: Classes of algorithms for achieving particular evaluation criteria in a certain set

Note: We will see that some approaches can achieve multiple frameworks in multiple settings
Table of Contents

1 MDPs

2 Bayesian MDPs

3 Generalization and Exploration

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

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

1: Given $\epsilon$, $\delta$, $m$
2: $\beta = \frac{1}{1-\gamma} \sqrt{0.5 \ln(2 |S| |A| m/) \delta}$
3: $n_{sas}(s, a, s') = 0$ $s \in S$, $a \in A$, $s' \in S$
4: $rc(s, a) = 0$, $n_{sa}(s, a) = 0$, $\tilde{Q}(s, a) = 1/(1 - \gamma)$ $\forall$ $s \in S$, $a \in A$
5: $t = 0$, $s_t = s_{init}$
6: loop
7: $a_t = \arg \max_{a \in A} Q(s_t, a)$
8: Observe reward $r_t$ and state $s_{t+1}$
9: $n_{sa}(s_t, a_t) = n(s_t, a_t) + 1$, $n_{sas}(s_t, a_t, s_{t+1}) = n_{sas}(s_t, a_t, s_{t+1}) + 1$
10: $rc(s_t, a_t) = \frac{rc(s_t, a_t)n_{sa}(s_t, a_t) + r_t}{(n_{sa}(s_t, a_t) + 1)}$
11: $\hat{R}(s, a) = \frac{rc(s_t, a_t)}{n(s_t, a_t)}$ and $\hat{T}(s'|s, a) = \frac{n_{sas}(s_t, a_t, s')}{n_{sa}(s_t, a_t)}$ $\forall s' \in S$
12: while not converged do
13: $\tilde{Q}(s, a) = \hat{R}(s, a) + \gamma \sum_{s'} \hat{T}(s'|s, a) \max_{a'} \tilde{Q}(s', a') + \frac{\beta}{\sqrt{n_{sa}(s, a)}}$ $\forall$ $s \in S$, $a \in A$
14: end while
15: end loop
For a given $\epsilon$ and $\delta$, 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 $\epsilon$-close to the optimal action, where $N$ is a polynomial function of $(|S|, |A|, \gamma, \epsilon, \delta)$.

Is this true for all algorithms?
Theorem 2. Suppose that $\epsilon$ and $\delta$ are two real numbers between 0 and 1 and $M = \langle S, A, T, R, \gamma \rangle$ is any MDP. There exists an input $m = m(\frac{1}{\epsilon}, \frac{1}{\delta})$, satisfying $m(\frac{1}{\epsilon}, \frac{1}{\delta}) = O\left(\frac{|S|}{\epsilon^2(1-\gamma)^4} + \frac{1}{\epsilon^2(1-\gamma)^4} \ln \frac{|S| |A|}{\epsilon(1-\gamma)^\delta}\right)$, 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 $A_t$ denote MBIE-EB’s policy at time $t$ and $s_t$ denote the state at time $t$. With probability at least $1 - \delta$, $V_{A_t}^M(s_t) \geq V^*_M(s_t) - \epsilon$ is true for all but $O\left(\frac{|S| |A|}{\epsilon^3(1-\gamma)^6} (|S| + \ln \frac{|S| |A|}{\epsilon(1-\gamma)^\delta}) \ln \frac{1}{\delta} \ln \frac{1}{\epsilon(1-\gamma)}\right)$ timesteps $t$. 
A Sufficient Set of Conditions to Make a RL Algorithm PAC

1. Optimism

\[ Q_t(s, a) = Q^*(s, a) - \varepsilon \quad \forall s, a, t \]

computed value should be optimistic with respect to real Q values

2. Accuracy

\[ V^\pi_t(s) - V^\pi(s) \leq \varepsilon \]

will define further. MDP related to true MDP will be denoted by $s_m$ MDP defined in MBE-ER

3) Bounded learning complexity:
- total # of updates to $Q$
- # times visit an "unknown" (s,a) pair bounded by $s(c, \delta)$

A RL alg that is executed on any MDP $M$ will follow a $O(c)$-optimal policy on all steps but on

\[ O\left( s(c, \delta) / \varepsilon (1 - \gamma)^2 \ln \frac{1}{\varepsilon} \ln \frac{1}{1 - 2s} \right) \]
Proof MBIE-EB is optimistic

MBIE-EB uses value iteration

1st compute a Bellman backup with empirical $V^*$

- for some $(s, a)$ consider it has been experienced $n(s, a) < m$

Let $x_i = r_i + \gamma V^*(s_i)$ where $r_i, s_i$ are the $i$th reward and next state reached on the $i$th time action $a$ was taken in state $s$

Note $E[x_i] = Q^*(s, a)$

$0 \leq x_i \leq \frac{1}{1-\gamma} \forall i = 1 \ldots n(s, a)$

one can use hoeffding's inequality (really should do martingale)

$\Pr \left[ E[x_i] - \frac{1}{n(s, a)} \sum_{i=1}^{n(s, a)} x_i - \gamma \sqrt{\ln(n(s, a))} \right] \leq e^{-2\beta^2(1-\gamma)^2}$

plug in $\beta = \frac{1}{1-\gamma} \sqrt{\frac{\ln(215111A1m)}{2}}$

after $n(s, a) = m$ stop changing $\hat{R}(s, a)$ and $\hat{T}(s, a)$

So using union bound

$\forall s, a \sum_{s'} \hat{T}(s', s, a)V^*(s') - Q^*(s, a) \geq -\frac{8}{\gamma} \sqrt{\ln(n(s, a))}$

with $\forall t \gamma^t u_t \leq \frac{8}{\gamma}$

now do proof by induction

do on board
How Does MBIE-EB Fulfill these Conditions?

Big idea in many analyses of optimism under uncertainty known set
(s,a) pairs where n(s,a) is large
intuitively should have good empirical estimate
of these pairs

s, a else
Table of Contents

1 MDPs

2 Bayesian MDPs

3 Generalization and Exploration

4 Summary
**Bayesian bandits** exploit prior knowledge of rewards, $p[R]$

They compute posterior distribution of rewards $p[R | h_t]$, 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
Consider a bandit problem where the reward of an arm is a binary outcome \( \{0, 1\} \) sampled from a Bernoulli with parameter \( \theta \)

- E.g. Advertisement click through rate, patient treatment succeeds/fails, ...

The Beta distribution \( \text{Beta}(\alpha, \beta) \) 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 family.

- Assume the prior over \( \theta \) is a \( \text{Beta}(\alpha, \beta) \) as above
- Then after observed a reward \( r \in \{0, 1\} \) then updated posterior over \( \theta \) is \( \text{Beta}(r + \alpha, 1 - r + \beta) \)
Thompson Sampling for Bandits

1: Initialize prior over each arm $a$, $p(\mathcal{R}_a)$
2: loop
3: For each arm $a$ sample a reward distribution $\mathcal{R}_a$ from posterior
4: Compute action-value function $Q(a) = \mathbb{E}[\mathcal{R}_a]$
5: $a_t = \arg \max_{a \in A} Q(a)$
6: Observe reward $r$
7: Update posterior $p(\mathcal{R}_a | r)$ using Bayes law
8: end loop
Bayesian Model-Based RL

- Maintain posterior distribution over MDP models
- Estimate both transition and rewards, \( p[\mathcal{P}, R \mid h_t] \), 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
  \[ \pi(s, a \mid h_t) = \mathbb{P}[Q(s, a) > Q(s, a'), \forall a' \neq a \mid h_t] \]
  \[ = \mathbb{E}_{\mathcal{P}, \mathcal{R}|h_t} \left[ 1(a = \arg\max_{a \in \mathcal{A}} Q(s, a)) \right] \]

- 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

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

Tabular (Final State A)

Emma Brunskill (CS234 Reinforcement Learning) Lecture 13: Fast Reinforcement Learning

Ben Van Roy’s group
<table>
<thead>
<tr>
<th></th>
<th>Table of Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MDPs</td>
</tr>
<tr>
<td>2</td>
<td>Bayesian MDPs</td>
</tr>
<tr>
<td>3</td>
<td>Generalization and Exploration</td>
</tr>
<tr>
<td>4</td>
<td>Summary</td>
</tr>
</tbody>
</table>
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)

1: Given $\epsilon$, $\delta$, $m$
2: $\beta = \frac{1}{1-\gamma} \sqrt{0.5 \ln(2|S||A|m/\delta}$
3: $n_{sas}(s, a, s') = 0 \ s \in S, \ a \in A, \ s' \in S$
4: $rc(s, a) = 0, \ n_{sa}(s, a) = 0, \ \tilde{Q}(s, a) = 1/(1-\gamma) \ \forall \ s \in S, \ a \in A$
5: $t = 0, \ s_t = s_{init}$
6: loop
7: $a_t = \arg \max_{a \in A} Q(s_t, a)$
8: Observe reward $r_t$ and state $s_{t+1}$
9: $n_{sa}(s_t, a_t) = n(s_t, a_t) + 1, \ n_{sas}(s_t, a_t, s_{t+1}) = n_{sas}(s_t, a_t, s_{t+1}) + 1$
10: $rc(s_t, a_t) = \frac{rc(s_t, a_t)n_{sa}(s_t, a_t) + r_t}{(n_{sa}(s_t, a_t) + 1)}$
11: $\hat{R}(s, a) = \frac{rc(s_t, a_t)}{n(s_t, a_t)}$ and $\hat{T}(s' | s, a) = \frac{n_{sas}(s_t, a_t, s')}{n_{sa}(s_t, a_t)} \ \forall s' \in S$
12: while not converged do
13: $\tilde{Q}(s, a) = \hat{R}(s, a) + \gamma \sum_{s'} \hat{T}(s' | s, a) \max_{a'} \tilde{Q}(s', a') + \frac{\beta}{\sqrt{n_{sa}(s, a)}} \ \forall s \in S, \ a \in A$
14: end while
15: 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 \( r + \gamma \max_a \hat{Q}(s', a'; w) \) which leverages the max of the current function approximation value

\[
\Delta w = \alpha (r(s) + \gamma \max_{a'} \hat{Q}(s', a'; \overline{w}) - \hat{Q}(s, a; w)) \nabla_w \hat{Q}(s, a; w)
\]
Recall: Value Function Approximation with Control

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

$$
\Delta w = \alpha (r(s) + r_{\text{bonus}}(s, a) + \gamma \max_{a'} \hat{Q}(s', a'; \bar{w}) - \hat{Q}(s, a; w)) \nabla_w \hat{Q}(s, a; w)
$$
For Q-learning use a TD target $r + \gamma \max_a \hat{Q}(s', a'; w)$ which leverages the max of the current function approximation value

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

$r_{\text{bonus}}(s, a)$ 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

Bellemare et al. “Unifying Count-Based Exploration and Intrinsic Motivation”

Enormously better than standard DQN with $\epsilon$-greedy approach

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.
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
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

1 MDPs

2 Bayesian MDPs

3 Generalization and Exploration

4 Summary
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 RL
- **This time:** Fast RL
- Next time: Meta-learning