The prior over arm 1 is Beta(1,2) (left) and arm 2 is a Beta(1,1) (right). 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 parameter 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
Last time: Fast Learning (Bayesian bandits to MDPs)

This time: Fast Learning III (MDPs)

Next time: Batch RL
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 (PAC)

- **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>
<thead>
<tr>
<th></th>
<th>Topic</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>
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
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)

1: Given $\epsilon$, $\delta$, $m$
2: $\beta = \frac{1}{2\gamma} \sqrt{\frac{\ln(2|S||A|m/\delta)}{n_{sa}(0)}}$
3: $n_{sas}(s, a, s') = 0, \forall s \in S, a \in A, s' \in S$
4: $r_c(s, a) = 0, \tilde{Q}(s, a) = \frac{1}{1-(1-\gamma)} \forall s, a$
5: $t = 0, s_t = s_{init}$
6: loop
7: $a_t = \arg\max_a \tilde{Q}(s, a)$
8: observe reward $r_t$ and $s_{t+1}$
9: $n_{sas} \leftarrow n_{sas} + 1$
10: $r_c = (r_c \cdot (n_{sa}-1) + r_t) / n_{sa}$ empirical mean
11: $\hat{T}(s'|s, a) = n_{sas} \tilde{Q}(s', a, s') / n_{sa}(s, a)$
12: while not converged do
13: $\tilde{Q}(s, a) = r_c(s, a) + \gamma \sum_{s'} \tilde{T}(s'|s, a) \max_a \tilde{Q}(s', a')$
14: end while
15: end loop
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: $n_{sas}(s, a, s') = 0, \forall s \in S, a \in A, s' \in S$
4: $rc(s, a) = 0, n_{sa}(s, a) = 0, \bar{Q}(s, a) = 1/(1 - \gamma), \forall s \in S, a \in A$
5: $t = 0, s_t = s_{init}$
6: loop
7: 
8: 
9: 
10: 
11: 
12: while not converged do
13: 
14: end while
15: end loop
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: $n_{sas}(s, a, s') = 0$, $\forall s \in S$, $a \in A$, $s' \in S$
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: $a_t = \arg \max_{a \in A} \tilde{Q}(s_t, a)$
8: Observe reward $r_t$ and state $s_{t+1}$
9: $n_{sa}(s_t, a_t) = n_{sa}(s_t, a_t) + 1$, $n_{sas}(s_t, a_t, s_{t+1}) = n_{sas}(s_t, a_t, s_{t+1}) + 1$
10: $rc(s_t, a_t) = \frac{rc(s_t, a_t)(n_{sa}(s_t, a_t)-1)+r_t}{n_{sa}(s_t, a_t)}$
11: $\hat{R}(s_t, a_t) = rc(s_t, a_t)$ and $\hat{T}(s'|s_t, a_t) = \frac{n_{sas}(s_t, a_t, s'}{n_{sa}(s_t, a_t)}$, $\forall s' \in S$
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

Update each $s, a$ at most $m$ times

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

**No**
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 $\beta = \left(\frac{1}{1-\gamma}\right)\sqrt{\ln(2S|A|m/\delta)}/2$ 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 $1 - \delta$, $V_{M}^{A_t}(s_t) \geq V_{M}^{\ast}(s_t) - \epsilon$ 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 $t$. 

$\epsilon = .1 \quad \gamma = .9$
A Sufficient Set of Conditions to Make a RL Algorithm

PAC

1. Optimism

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

\[ Q_t = \frac{1}{(1-\gamma)} \]

2. Accuracy

\[ V_t(s) - V_\pi^t(s) \leq \epsilon \]

"Known set" of MDP related to the MDP $s$ MDP defined in MBE-EB

"Known set" of MDP sampled enough that have a good enough estimate of its reward dynamics
3) Bounded learning complexity

\[ \sum_{t} \text{total # of updates to } Q \]
\[ \text{# times we visit an unknown } (s,a) \text{ pair} \]
\[ C_{1}(e, s) \]

A RL algorithm that is executed on any MDP will follow an \( O(\epsilon) \)-optimal policy on all but \( N \) steps.

\[
N = O \left( \frac{C_{1}(e, s)}{\epsilon (1-\gamma)^2} \ln \frac{1}{\delta} \ln \frac{1}{\epsilon(1-\gamma)} \right)
\]

needs to be

\[ w/\text{prob } \geq 1 - 2\delta \]

poly in \( S, A, \text{ etc.} \)

for RL alg to be PAC
How Does MBIE-EB Fulfill these Conditions?
# Table of Contents

1. MDPs

2. Bayesian MDPs

3. Generalization and Exploration

4. Summary
Bayesian bandits exploit prior knowledge of rewards, $p[\mathcal{R}]$

They compute posterior distribution of rewards $p[\mathcal{R} | h_t]$, 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
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 \( \theta \).

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

The Beta distribution \( \text{Beta}(\alpha, \beta) \) is conjugate for the Bernoulli distribution

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

where \( \Gamma(x) \) is the Gamma function.

- 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. $a_t = \arg \max_{a \in \mathcal{A}} Q(a)$
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, $p[\mathcal{P}, \mathcal{R} \mid h_t]$, 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

\[
\pi(s, a \mid h_t) = \mathbb{P}[Q(s, a) \geq Q(s, a'), \forall a' \neq a \mid h_t]
\]

\[
= \mathbb{E}_{\mathcal{P}, \mathcal{R} \mid h_t} \left[ \mathbb{1}(a = \arg \max_{a \in A} Q(s, a)) \right]
\]

- 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, \( a_t = \arg \max_{a \in A} Q^*(s_t, a) \)

\[\forall s, a \sim \mathcal{P}(r_{s,a} = \theta) = \text{Beta} \ldots\]
Thompson Sampling for MDPs

1: Initialize prior over the dynamics and reward models for each \((s, a)\),
   \(p(R_{as}), p(T(s'|s, a))\)   ex. Beta, Dirichlet
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**
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
   - F
2. Can involve using optimism with respect to both the possible dynamics and reward models in order to compute an optimistic Q function
   - T
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).
   - F
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
   - F
2. Must perform MDP planning every time the posterior is updated
   - T
3. Has the same computational cost each step as Q-learning
   - T
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-wm0Pkl&feature=youtu.be
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?
1: Given $\epsilon$, $\delta$, $m$

2: $\beta = \frac{1}{1-\gamma} \sqrt{0.5 \ln(2|S||A|m/\delta)}$

3: $n_{sas}(s, a, s') = 0$, $\forall s \in S$, $a \in A$, $s' \in S$

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: $a_t = \arg \max_{a \in A} \tilde{Q}(s_t, a)$

8: Observe reward $r_t$ and state $s_{t+1}$

9: $n_{sa}(s_t, a_t) = n_{sa}(s_t, a_t) + 1$, $n_{sas}(s_t, a_t, s_{t+1}) = n_{sas}(s_t, a_t, s_{t+1}) + 1$

10: $rc(s_t, a_t) = \frac{rc(s_t, a_t)(n_{sa}(s_t, a_t) - 1) + r_t}{n_{sa}(s_t, a_t)}$

11: $\hat{R}(s_t, a_t) = rc(s_t, a_t)$ and $\hat{T}(s'|s_t, a_t) = \frac{n_{sas}(s_t, a_t, s')}{n_{sa}(s_t, a_t)}$, $\forall s' \in S$

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
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

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

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

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

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

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

Enormously better than standard DQN with $\epsilon$-greedy approach
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
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
Last time: Fast Learning (Bayesian bandits to MDPs)
This time: Fast Learning III (MDPs)
Next time: Batch RL