Class Structure

- Last time: Midterm!
- **This time:** Exploration and Exploitation
- Next time: Batch RL
Atari: Focus on the $x$-axis

- Last time: Midterm!
- **This time:** Exploration and Exploitation
- Next time: Batch RL
Asymptotic convergence to good/optimal is not enough
Table of Contents

1. Metrics for evaluating RL algorithms
2. Exploration and Exploitation
3. Principles for RL Exploration
4. Multi-Armed Bandits
5. MDPs
6. Principles for RL Exploration
Performance Criteria of RL Algorithms

- Empirical performance
- Convergence (to something ...)
- Asymptotic convergence to optimal policy
- Finite sample guarantees: probably approximately correct
- Regret (with respect to optimal decisions)
- Optimal decisions given information have available
- PAC uniform
Performance Criteria of RL Algorithms

- **Empirical performance**
- **Convergence** (to something ...)
- **Asymptotic convergence to optimal policy**
- Finite sample guarantees: probably approximately correct
- **Regret** (with respect to optimal decisions)
- Optimal decisions given information have available
- **PAC uniform**
Strategic Exploration

- To get stronger guarantees on performance, need **strategic exploration**
Table of Contents

1. Metrics for evaluating RL algorithms
2. Exploration and Exploitation
3. Principles for RL Exploration
4. Multi-Armed Bandits
5. MDPs
6. Principles for RL Exploration
Online decision-making involves a fundamental choice:

- **Exploitation**: Make the best decision given current information
- **Exploration**: Gather more information

The best long-term strategy may involve short-term sacrifices

Gather enough information to make the best overall decision
Examples

- Restaurant Selection
  - Go off-campus
  - Eat at Treehouse (again)
- Online advertisements
  - Show the most successful ad
  - Show a different ad
- Oil Drilling
  - Drill at best known location
  - Drill at new location
- Game Playing
  - Play the move you believe is best
  - Play an experimental move
Principles

- Naive Exploration
- Optimistic Initialization
- Optimism in the Face of Uncertainty
- Probability Matching
- Information State Search
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Metrics for evaluating RL algorithms</td>
</tr>
<tr>
<td>2</td>
<td>Exploration and Exploitation</td>
</tr>
<tr>
<td>3</td>
<td>Principles for RL Exploration</td>
</tr>
<tr>
<td>4</td>
<td>Multi-Armed Bandits</td>
</tr>
<tr>
<td>5</td>
<td>MDPs</td>
</tr>
<tr>
<td>6</td>
<td>Principles for RL Exploration</td>
</tr>
</tbody>
</table>
MABs

- Will introduce various principles for multi-armed bandits (MABs) first instead of for generic reinforcement learning
- MABs are a subclass of reinforcement learning
- Simpler (as will see shortly)
Multiarmed Bandits

- Multi-armed bandit is a tuple of \((\mathcal{A}, \mathcal{R})\)
- \(\mathcal{A}\) : known set of \(m\) actions
- \(\mathcal{R}^a(r) = \mathbb{P}[r | a]\) is an unknown probability distribution over rewards
- At each step \(t\) the agent selects an action \(a_t \in \mathcal{A}\)
- The environment generates a reward \(r_t \sim \mathcal{R}^{a_t}\)
- Goal: Maximize cumulative reward \(\sum_{\tau=1}^{t} r_\tau\)
Greedy Algorithm

- We consider algorithms that estimate $\hat{Q}_t(a) \approx Q(a)$
- Estimate the value of each action by Monte-Carlo evaluation

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t=1}^{T} r_t 1(a_t = a)$$

- The greedy algorithm selects action with highest value

$$a^*_t = \arg \max_{a \in A} \hat{Q}_t(a)$$

- Greedy can lock onto suboptimal action, forever
\( \epsilon \)-Greedy Algorithm

- With probability \( 1 - \epsilon \) select \( a = \arg \max_{a \in A} \hat{Q}_t(a) \)
- With probability \( \epsilon \) select a random action

- Always will be making a sub-optimal decision \( \epsilon \) fraction of the time
- Already used this in prior homeworks
Optimistic Initialization

- Simple and practical idea: initialize $Q(a)$ to high value
- Update action value by incremental Monte-Carlo evaluation
- Starting with $N(a) > 0$

$$
\hat{Q}_t(a_t) = \hat{Q}_{t-1} + \frac{1}{N_t(a_t)}(r_t - \hat{Q}_{t-1})
$$

- Encourages systematic exploration early on
- But can still lock onto suboptimal action$^{21}$

---

$^{21}$ Depends on how high initialize $Q$
Decaying $\epsilon_t$-Greedy Algorithm

- Pick a decay schedule for $\epsilon_1, \epsilon_2, \ldots$
- Consider the following schedule

\[
c > 0
\]
\[
d = \min_{a : \Delta_a > 0} \Delta_i
\]
\[
\epsilon_t = \min \left\{ 1, \frac{c|A|}{d^2 t} \right\}
\]
How to Compare these Methods?

- Empirical performance
- Convergence (to something ...)
- Asymptotic convergence to optimal policy
- Finite sample guarantees: probably approximately correct
- **Regret (with respect to optimal decisions)**
  - Very common criteria for bandit algorithms
  - Also frequently considered for reinforcement learning methods
- Optimal decisions given information have available
- PAC uniform
Regret

- **Action-value** is the mean reward for action $a$

  $$Q(a) = \mathbb{E}[r \mid a]$$

- **Optimal value $V^*$**

  $$V^* = Q(a^*) = \max_{a \in A} Q(a)$$

- **Regret** is the opportunity loss for one step

  $$l_t = \mathbb{E}[V^* - Q(a_t)]$$

- **Total Regret** is the total opportunity loss

  $$L_t = \mathbb{E}\left[\sum_{\tau=1}^{t} V^* - Q(a_\tau)\right]$$

- Maximize cumulative reward $\iff$ minimize total regret
Evaluating Regret

- **Count** $N_t(a)$ is expected number of selections for action $a$
- **Gap** $\Delta_a$ is the difference in value between action $a$ and optimal action $a^*$, $\Delta_a = V^* - Q(a)$
- Regret is a function of gaps and counts

$$L_t = \mathbb{E} \left[ \sum_{\tau=1}^{t} V^* - Q(a_{\tau}) \right]$$

$$= \sum_{a \in A} \mathbb{E}[N_t(a)](V^* - Q(a))$$

$$= \sum_{a \in A} \mathbb{E}[N_t(a)]\Delta_a$$

- A good algorithm ensures small counts for large gaps
- But: gaps are not known
Types of Regret bounds

- **Problem independent**: Bound how regret grows as a function of $T$, the total number of time steps the algorithm operates for.
- **Problem dependent**: Bound regret as a function of the number of times pull each arm and the gap between the reward for the pulled arm and the true optimal arm.
"Good": Sublinear or below regret

- **Explore forever**: have linear total regret
- **Explore never**: have linear total regret
- Is it possible to achieve sublinear regret?
Greedy Bandit Algorithms and Optimistic Initialization

- **Greedy**: Linear total regret
- **Constant \( \epsilon \)-greedy**: Linear total regret
- **Decaying \( \epsilon \)-greedy**: Sublinear regret but schedule for decaying \( \epsilon \) requires knowledge of gaps, which are unknown
- **Optimistic initialization**: Sublinear regret if initialize values sufficiently optimistically, else linear regret
- Check your understanding: why does fixed \( \epsilon \)-greedy have linear regret? (Do a proof sketch)
Lower Bound

- Use lower bound to determine how hard this problem is.
- The performance of any algorithm is determined by similarity between optimal arm and other arms.
- Hard problems have similar looking arms with different means.
- This is described formally by the gap $\Delta_a$ and the similarity in distributions $KL(\mathcal{R}^a || \mathcal{R}^{a*})$.
- Theorem (Lai and Robbins): Asymptotic total regret is at least logarithmic in number of steps.

$$\lim_{t \to \infty} L_t \geq \log t \sum_{a \mid \Delta_a > 0} \frac{\Delta_a}{KL(\mathcal{R}^a || \mathcal{R}^{a*})}$$
Principles

- Naive Exploration
- Optimistic Initialization
- Optimism in the Face of Uncertainty
- Probability Matching
- Information State Search
Optimism in the Face of Uncertainty

Which action should we pick?
Choose arms that could be good
Intuitively choosing an arm with potentially high mean reward will either lead to:
- Getting high reward: if the arm really has a high mean reward
- Learn something: if the arm really has a lower mean reward, pulling it will (in expectation) reduce its average reward and the uncertainty over its value
Upper Confidence Bounds

- Estimate an upper confidence $\hat{U}_t(a)$ for each action value, such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with high probability.
- This depends on the number of times $N_t(a)$ has been selected:
  - Small $N_t(a) \rightarrow$ large $\hat{U}_t(a)$ (estimate value is uncertain)
  - Large $N_t(a) \rightarrow$ small $\hat{U}_t(a)$ (estimate value is accurate)
- Select action maximizing Upper Confidence Bound (UCB)

$$a_t = \arg \max_{a \in A} \hat{Q}_t(a) + \hat{U}_t(a)$$
Hoeffding’s Inequality

- **Theorem (Hoeffding’s Inequality):** Let \( X_1, \ldots, X_t \) be i.i.d. random variables in \([0, 1]\), and let \( \bar{X}_t = \frac{1}{t} \sum_{\tau=1}^{t} X_\tau \) be the sample mean. Then

\[
P[\mathbb{E}[X] > \bar{X}_t + u] \leq \exp(-2tu^2)
\]

- **Applying Hoeffding’s Inequality to the rewards of the bandit,**

\[
P\left[ Q(a) > \hat{Q}_t(a) + U_t(a) \right] \leq \exp(-2N_t(a)U_t(a)^2)
\]
Calculating UCB

- Pick a probability $p$ that true value exceeds UCB
- Now solve for $U_t(a)$

$$\exp(-2N_t(a)U_t(a)^2) = p$$

$$U_t(a) = \sqrt{-\log p \over 2N_t(a)}$$

- Reduce $p$ as we observe more rewards, e.q. $p = t^{-4}$
- Ensures we select optimal action as $t \to \infty$

$$U_t(a) = \sqrt{2 \log t \over N_t(a)}$$
This leads to the UCB1 algorithm

\[ a_t = \arg \max_{a \in A} Q(a) + \sqrt{2 \log t N_t(a)} \]

Theorem: The UCB algorithm achieves logarithmic asymptotic total regret

\[ \lim_{t \to \infty} L_t \leq 8 \log t \sum_{a \mid \Delta_a > 0} \Delta_a \]
An alternative would be to always select the arm with the highest lower bound. Why can this yield linear regret?
Bayesian Bandits

- So far we have made no assumptions about the reward distribution $\mathcal{R}$
  - Except bounds on rewards
- **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
Assume reward distribution is Gaussian, $\mathcal{R}_a(r) = \mathcal{N}(r; \mu_a, \sigma^2_a)$.

Compute Gaussian posterior over $\mu_a$ and $\sigma^2_a$ (by Bayes law):

$$p[\mu_a, \sigma^2_a | h_t] \propto p[\mu_a, \sigma^2_a] \prod_{t \mid a_t = a} \mathcal{N}(r_t; \mu_a, \sigma^2_a)$$

Pick action that maximizes standard deviation of $Q(a)$:

$$a_t = \arg\max_{a \in A} \mu_a + c \frac{c \sigma_a}{\sqrt{N(a)}}$$
Principles

- Naive Exploration
- Optimistic Initialization
- Optimism in the Face of Uncertainty
- **Probability Matching**
- Information State Search
Probability Matching

- Again assume have a parametric distribution over rewards for each arm
- **Probability matching** selects action $a$ according to probability that $a$ is the optimal action

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

- Probability matching is optimistic in the face of uncertainty
  - Uncertain actions have higher probability of being max
- Can be difficult to compute analytically from posterior
- **Thompson sampling** implements probability matching

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

\[
= \mathbb{E}_{R \mid h_t} \left[ 1(a = \text{arg max}_{a \in \mathcal{A}} Q(a)) \right]
\]

- Use Bayes law to compute posterior distribution \( p[R \mid h_t] \)
- **Sample** a reward distribution \( R \) from posterior
- Compute action-value function \( Q(a) = \mathbb{E}[R_a] \)
- Select action maximizing value on sample, \( a_t = \text{arg max}_{a \in \mathcal{A}} Q(a) \)
- Thompson sampling achieves Lai and Robbins lower bound
- Last checked: bounds for optimism are tighter than for Thompson sampling
- But empirically Thompson sampling can be extremely effective
Principles

- Naive Exploration
- Optimistic Initialization
- Optimism in the Face of Uncertainty
- Probability Matching
- Information State Search
Exploration is useful because it gains information

Can we quantify the **value of information (VOI)**?

- How much reward a decision-maker would be prepared to pay in order to have that information, prior to making a decision
- Long-term reward after getting information - immediate reward
Consider bandit where only get to make a **single** decision

Oil company considering buying rights to drill in 1 of 5 locations

1 of locations contains $10 million worth of oil, others 0

Cost of buying rights to drill is $2 million

Seismologist says for a fee will survey one of 5 locations and report back definitively whether that location does or does not contain oil

What is the should consider paying seismologist?
1 of locations contains $10$ million worth of oil, others $0$
Cost of buying rights to drill is $2$ million
Seismologist says for a fee will survey one of 5 locations and report back definitively whether that location does or does not contain oil
Value of information: expected profit if ask seismologist minus expected profit if don’t ask
Expected profit if don’t ask:
- Guess at random
  \[
  \frac{1}{5}(10 - 2) + \frac{4}{5}(0 - 2) = 0
  \]  

Expected profit if ask:
- If one surveyed has oil, expected profit is: \( 10 - 2 = 8 \)
- If one surveyed doesn’t have oil, expected profit: (guess at random from other locations) \( \frac{1}{4}(10 - 2) - \frac{3}{4}(-2) = 0.5 \)
- Weigh by probability will survey location with oil: \( \frac{1}{5} \times 8 + \frac{4}{5} \times 0.5 = 2 \)

\[ \text{VOI: } 2 - 0 = 2 \]
Relevant Background: Value of Information

- Back to making a sequence of decisions under uncertainty
- Information gain is higher in uncertain situations
- But need to consider value of that information
  - Would it change our decisions?
  - Expected utility benefit
So far viewed bandits as a simple fully observable Markov decision process (where actions don’t impact next state)

Beautiful idea: frame bandits as a partially observable Markov decision process where the hidden state is the mean reward of each arm
So far viewed bandits as a simple fully observable Markov decision process (where actions don’t impact next state)

Beautiful idea: frame bandits as a partially observable Markov decision process where the hidden state is the mean reward of each arm

(Hidden) State is static

Actions are same as before, pulling an arm

Observations: Sample from reward model given hidden state

POMDP planning $\Rightarrow$ Optimal Bandit learning
Information State Space

- POMDP belief state / information state $\tilde{s}$ is posterior over hidden parameters (e.g. mean reward of each arm)
- $\tilde{s}$ is a statistic of the history, $\tilde{s} = f(h_t)$
- Each action $a$ causes a transition to a new information state $\tilde{s}'$ (by adding information), with probability $\tilde{P}_{\tilde{s},\tilde{s}'}^a$
- Equivalent to a POMDP
- Or a MDP $\tilde{\mathcal{M}} = (\tilde{S}, \tilde{A}, \tilde{P}, \tilde{R}, \gamma)$ in augmented information state space
Consider a Bernoulli bandit such that $\mathcal{R}^a = \mathcal{B}(\mu_a)$
e.g. Win or lose a game with probability $\mu_a$
Want to find which arm has the highest $\mu_a$
The information state is $\tilde{s} = (\alpha, \beta)$
- $\alpha_a$ counts the pulls of arm $a$ where the reward was 0
- $\beta_a$ counts the pulls of arm $a$ where the reward was 1
Solving Information State Space Bandits

- We now have an infinite MDP over information states.
- This MDP can be solved by reinforcement learning.
- Model-free reinforcement learning (e.g. Q-learning).
- Bayesian model-based RL (e.g. Gittins indices).
- This approach is known as Bayes-adaptive RL: Finds Bayes-optimal exploration/exploitation trade-off with respect to prior distribution.
- In other words, selects actions that maximize expected reward given information have so far.
- Check your understanding: Can an algorithm that optimally solves an information state bandit have a non-zero regret? Why or why not?
Bayes-Adaptive Bernoulli Bandits

- Start with $\text{Beta}(\alpha_a, \beta_a)$ prior over reward function $R^a$
- Each time $a$ is selected, update posterior for $R^a$
  - $\text{Beta}(\alpha_a + 1, \beta_a)$ if $r = 0$
  - $\text{Beta}(\alpha_a, \beta_a + 1)$ if $r = 1$
- This defines transition function $\tilde{P}$ for the Bayes-adaptive MDP
- Information state $(\alpha, \beta)$ corresponds to reward model $\text{Beta}(\alpha, \beta)$
- Each state transition corresponds to a Bayesian model update
$s_t^+ = 1: \begin{bmatrix} \alpha_1^1 & \beta_1^1 & \alpha_2^2 & \beta_2^2 \\ \alpha_2^1 & \beta_2^1 & \alpha_1^2 & \beta_1^2 \end{bmatrix}$

$s_t^+ = 1: \begin{bmatrix} \alpha_1^1 & \beta_1^1 & \alpha_2^2 & \beta_2^2 \\ \alpha_2^1 & \beta_2^1 & \alpha_1^2 & \beta_1^2 \end{bmatrix}$

$s_t^+ = 2: \begin{bmatrix} \alpha_1^1 (\beta_1^1 + 1) & \alpha_2^2 & \beta_2^2 \\ \alpha_2^1 & \beta_2^1 & \alpha_1^2 \end{bmatrix}$

$s_t^+ = 2: \begin{bmatrix} \alpha_1^1 (\beta_1^1 + 1) & \alpha_2^2 & \beta_2^2 \\ \alpha_2^1 & \beta_2^1 & \alpha_1^2 \end{bmatrix}$

$s_t^+ = 1: \begin{bmatrix} \alpha_1^1 & \beta_1^1 & \alpha_2^2 (\beta_2^1 + 1) \\ \alpha_2^1 & \beta_2^1 & \alpha_1^2 \end{bmatrix}$

$s_t^+ = 2: \begin{bmatrix} \alpha_1^1 & \beta_1^1 & \alpha_2^2 (\beta_2^1 + 1) \\ \alpha_2^1 & \beta_2^1 & \alpha_1^2 \end{bmatrix}$
Bayes-adaptive MDP can be solved by dynamic programming

- The solution is known as the Gittins index
- Exact solution to Bayes-adaptive MDP is typically intractable; information state space is too large
- Recent idea: apply simulation-based search (Guez et al. 2012, 2013)
  - Forward search in information state space
  - Using simulations from current information state
Table of Contents

1. Metrics for evaluating RL algorithms
2. Exploration and Exploitation
3. Principles for RL Exploration
4. Multi-Armed Bandits
5. MDPs
6. Principles for RL Exploration
The sample principles for exploration/exploitation apply to MDPs

- Naive Exploration
- Optimistic Initialization
- Optimism in the Face of Uncertainty
- Probability Matching
- Information State Search
Optimistic Initialization: Model-Free RL

- Initialize action-value function $Q(s,a)$ to $\frac{r_{\text{max}}}{1-\gamma}$
- Run favorite model-free RL algorithm
  - Monte-Carlo control
  - Sarsa
  - Q-learning
  - etc.
- Encourages systematic exploration of states and actions
Construct an **optimistic** model of the MDP

- Initialize transitions to go to terminal state with $r_{max}$ reward
- Solve optimistic MDP by favorite planning algorithm
- Encourages systematic exploration of states and actions
- e.g. RMax algorithm (Brafman and Tennenholtz)
Maximize UCB on action-value function $Q^\pi(s, a)$

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

- Estimate uncertainty in policy evaluation (easy)
- Ignores uncertainty from policy improvement

Maximize UCB on optimal action-value function $Q^*(s, a)$

$$a_t = \arg \max_{a \in A} Q(s_t, a) + U_1(s_t, a) + U_2(s_t, a)$$

- Estimate uncertainty in policy evaluation (easy)
- plus uncertainty from policy improvement (hard)
Bayesian Model-Based RL

- Maintain posterior distribution over MDP models
- Estimate both transition and rewards, $p[\mathcal{P}, \mathcal{R} | h_t]$, 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 implements probability matching

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

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

- 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, \( a_t = \arg \max_{a \in \mathcal{A}} Q^*(s_t, a) \)
MDPs can be augmented to include information state

Now the augmented state is \((s, \tilde{s})\)

- where \(s\) is original state within MDP
- and \(\tilde{s}\) is a statistic of the history (accumulated information)

Each action \(a\) causes a transition

- to a new state \(s'\) with probability \(P_{s,s'}^a\)
- to a new information state \(\tilde{s}'\)

Defines MDP \(\tilde{M}\) in augmented information state space
Bayes Adaptive MDP

- Posterior distribution over MDP model is an information state
  \[ \tilde{s}_t = \mathbb{P}[\mathcal{P}, \mathcal{R} \mid h_t] \]

- Augmented MDP over \((s, \tilde{s})\) is called **Bayes-adaptive MDP**
- Solve this MDP to find optimal exploration/exploitation trade-off (with respect to prior)
- However, Bayes-adaptive MDP is typically enormous
- Simulation-based search has proven effective (Guez et al, 2012, 2013)
<table>
<thead>
<tr>
<th></th>
<th>Metrics for evaluating RL algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Exploration and Exploitation</td>
</tr>
<tr>
<td>3</td>
<td>Principles for RL Exploration</td>
</tr>
<tr>
<td>4</td>
<td>Multi-Armed Bandits</td>
</tr>
<tr>
<td>5</td>
<td>MDPs</td>
</tr>
<tr>
<td>6</td>
<td>Principles for RL Exploration</td>
</tr>
</tbody>
</table>
Principles

- **Naive Exploration**
  - Add noise to greedy policy (e.g. $\epsilon$-greedy)

- **Optimistic Initialization**
  - Assume the best until proven otherwise

- **Optimism in the Face of Uncertainty**
  - Prefer actions with uncertain values

- **Probability Matching**
  - Select actions according to probability they are best

- **Information State Search**
  - Lookahead search incorporating value of information
Other evaluation criteria

• Probably approximately correct:
  • On all but $N$ steps, algorithm will select an action whose value is near optimal $Q(s, a_t) - V(s) \geq -\epsilon$ with probability at least $1 - \delta$.
  • $N$ is a polynomial function of the MDP parameters ($|S|, |A|, \frac{1}{1-\gamma}, \delta, \epsilon$)

• Bounded "mistakes"

• Many PAC RL algorithms use ideas of optimism under uncertainty
Generalization and Strategic Exploration

- Significant interest in combining generalization with strategic exploration
- Many approaches are grounded by the principles outlined in this lecture
- Some examples:
  - Optimism under uncertainty: Bellemare et al. NIPS 2016; Ostrovski et al. ICML 2017; Tang et al. NIPS 2017
  - Probability matching: Osband et al. NIPS 2016; Mandel et al. IJCAI 2016
Last time: Midterm!

This time: Exploration and Exploitation

Next time: Batch RL