CSC415: Introduction to Reinforcement Learning

Lecture 7: Exploration in Deep RL

Dr. Amey Pore

Winter 2026

February 25, 2026

Structure adapted from A. Lazaric's lecture on Exploration-Exploitation (RLSS 2023), David Silver's course on RL

Lecture Outline

1. **Exploration Theory**
2. Course Logistics
3. Exploration in Deep RL

Exploration vs. Exploitation Dilemma

- 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 decisions

Examples

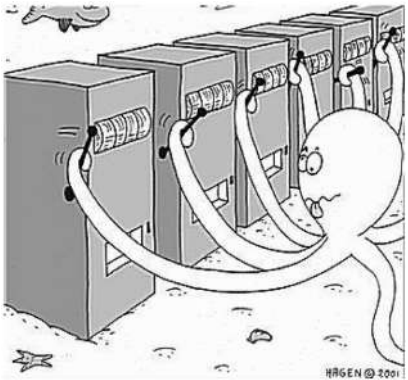
- **Restaurant Selection**
  - Exploitation: Go to your favourite restaurant
  - Exploration: Try a new restaurant
- **Online Banner Advertisements**
  - Exploitation: Show the most successful advert
  - Exploration: Show a different advert
- **Game Playing**
  - Exploitation: Play the move you believe is best
  - Exploration: Play an experimental move

Principles

- **Naive Exploration:** Add noise to greedy policy (e.g. ε-greedy)
- **Optimistic Initialisation:** 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

The Multi-Armed Bandit

- A multi-armed bandit is a tuple $\langle \mathcal{A}, \mathcal{R} \rangle$
  - $\mathcal{A}$ is a known set of m actions (or "arms")
  - $\mathcal{R}^a(r) = \mathbb{P}[r \mid a]$ 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 $r_t \sim \mathcal{R}^{a_t}$
- The goal is to maximise cumulative reward $\sum_{\tau=1}^{t} r_\tau$



Regret

- The action-value is the mean reward for action a,

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

- The optimal value V^* is

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

- The **regret** is the opportunity loss for one step

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

- The **total regret** is the total opportunity loss

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

- Maximise cumulative reward $\equiv$ minimise total regret

Counting Regret

- The count $N_t(a)$ is expected number of selections for action a
- The gap Δ_a is the difference in value between action a and optimal action a^*,

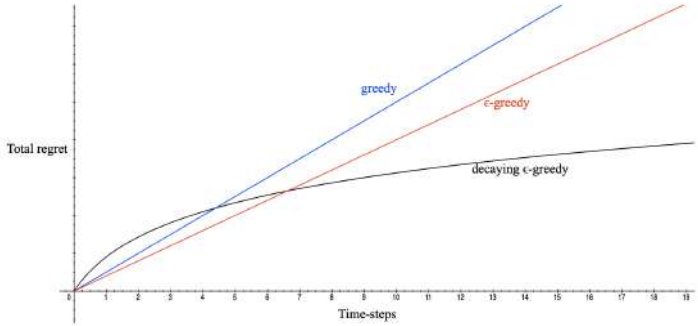
$$\Delta_a = V^* - Q(a)$$

- Regret is a function of gaps and the counts

$$L_t = \mathbb{E}\left[\sum_{\tau=1}^{t} V^* - Q(a_\tau)\right] = \sum_{a \in \mathcal{A}} \mathbb{E}[N_t(a)]\, \Delta_a$$

- A good algorithm ensures small counts for large gaps
- Problem: gaps are not known!

Linear or Sublinear Regret



- If an algorithm **forever explores** it will have linear total regret
- If an algorithm **never explores** it will have linear total regret
- Is it possible to achieve **sublinear** total regret?

Greedy Algorithm

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

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{\tau=1}^{T} r_\tau \mathbf{1}(a_\tau = a)$$

- The greedy algorithm selects action with highest value

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

- Greedy can lock onto a suboptimal action forever

$\Rightarrow$ Greedy has **linear** total regret

ε-Greedy Algorithm

- The ε-greedy algorithm continues to explore forever
  - With probability $1 - \varepsilon$ select $a = \arg\max_{a \in \mathcal{A}} \hat{Q}(a)$
  - With probability ε select a random action
- Constant ε ensures minimum regret

$$l_t \geq \frac{\varepsilon}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \Delta_a$$

$\Rightarrow$ ε-greedy has **linear** total regret

Decaying ε_t-Greedy Algorithm

- Pick a decay schedule for $\varepsilon_1, \varepsilon_2, \ldots$
- Consider the following schedule

$$
\begin{aligned}
c &> 0 \\
d &= \min_{a|\Delta_a > 0} \Delta_a \\
\varepsilon_t &= \min\left\{1, \frac{c|\mathcal{A}|}{d^2 t}\right\}
\end{aligned}
$$

- Decaying ε_t-greedy has **logarithmic** asymptotic total regret!
- Unfortunately, schedule requires advance knowledge of gaps
- **Goal:** find an algorithm with sublinear regret for any multi-armed bandit (without knowledge of $\mathcal{R}$)

Lower Bound

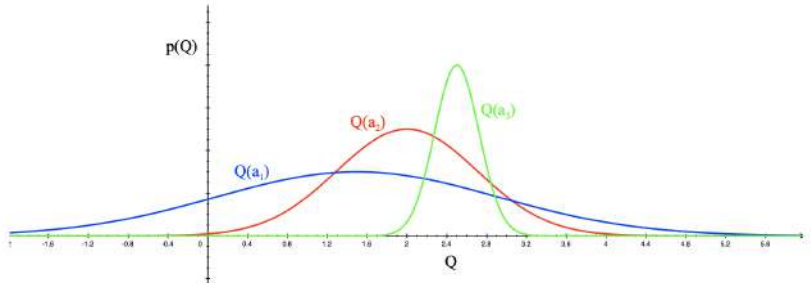
- The performance of any algorithm is determined by similarity between optimal arm and other arms
- Hard problems have similar-looking arms with different means
- This is described formally by the gap Δ_a and the similarity in distributions $\mathrm{KL}(\mathcal{R}^a \| \mathcal{R}^{a^*})$

Theorem (Lai and Robbins)

Asymptotic total regret is at least logarithmic in number of steps

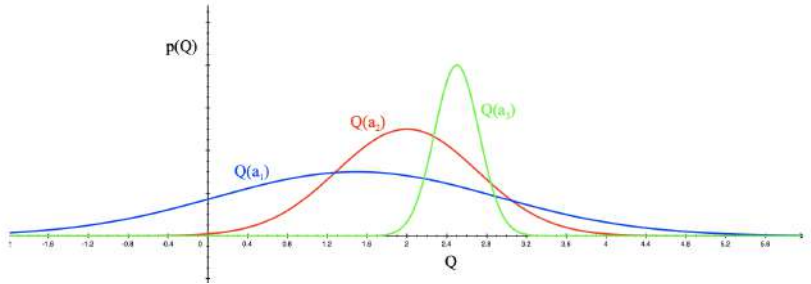
$$\lim_{t \to \infty} L_t \geq \log t \sum_{a | \Delta_a > 0} \frac{\Delta_a}{\mathrm{KL}(\mathcal{R}^a \| \mathcal{R}^{a^*})}$$

Optimism in the Face of Uncertainty



- Which action should we pick?
- The more **uncertain** we are about an action-value
- The more important it is to **explore** that action
- It could turn out to be the best action

Optimism in the Face of Uncertainty (2)



- After picking blue action
- We are less uncertain about the value
- And more likely to pick another action
- Until we home in on best action

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 $N(a)$ has been selected
  - Small $N_t(a) \Rightarrow$ large $\hat{U}_t(a)$ (estimated value is uncertain)
  - Large $N_t(a) \Rightarrow$ small $\hat{U}_t(a)$ (estimated value is accurate)
- Select action maximising Upper Confidence Bound (UCB)

$$a_t = \arg\max_{a \in \mathcal{A}} \hat{Q}_t(a) + \hat{U}_t(a)$$

Hoeffding's Inequality

Theorem (Hoeffding's Inequality)

Let $X_1, \ldots, X_t$ be i.i.d. random variables in $[0, 1]$, and let $\overline{X}_t = \frac{1}{t} \sum_{\tau=1}^{t} X_\tau$ be the sample mean. Then

$$\mathbb{P}\big[\mathbb{E}[X] > \overline{X}_t + u\big] \leq e^{-2tu^2}$$

- We apply Hoeffding's Inequality to rewards of the bandit conditioned on selecting action a

$$\mathbb{P}\big[Q(a) > \hat{Q}_t(a) + U_t(a)\big] \leq e^{-2N_t(a)U_t(a)^2}$$

Calculating Upper Confidence Bounds

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

$$e^{-2N_t(a)U_t(a)^2} = p$$

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

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

$$U_t(a) = \sqrt{\frac{2 \log t}{N_t(a)}}$$

UCB1

- This leads to the UCB1 algorithm

$$a_t = \arg\max_{a \in \mathcal{A}} Q(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

Theorem

The UCB algorithm achieves logarithmic asymptotic total regret

$$\lim_{t \to \infty} L_t \leq 8 \log t \sum_{a | \Delta_a > 0} \Delta_a$$

Example: UCB vs. ϵ-Greedy On 10-armed Bandit

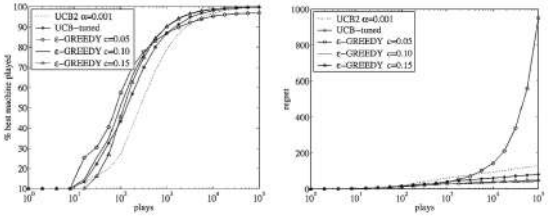


Figure 9. Comparison on distribution 11 (10 machines with parameters 0.9, 0.6, . . . , 0.6).

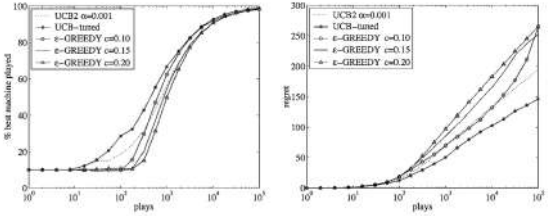
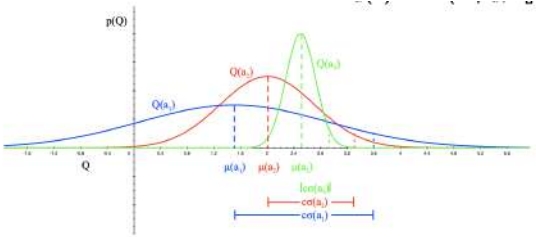


Figure 10. Comparison on distribution 12 (10 machines with parameters 0.9, 0.8, 0.8, 0.8, 0.7, 0.7, 0.7, 0.6, 0.6, 0.6).

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 $h_t = a_1, r_1, \ldots, a_{t-1}, r_{t-1}$ is the history
- Use posterior to guide exploration
  - Upper confidence bounds (Bayesian UCB)
  - Probability matching (Thompson sampling)
- Better performance if prior knowledge is accurate

Bayesian UCB Example: Independent Gaussians



- Assume reward is Gaussian: $\mathcal{R}_a(r) = \mathcal{N}(r; \mu_a, \sigma_a^2)$
- Posterior over μ_a, σ_a^2 (Bayes law):

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

- Pick action: $a_t = \arg\max_a \mu_a + c\,\sigma_a/\sqrt{N(a)}$

Probability Matching

- Probability matching selects action a according to probability that a is the optimal action

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

- 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

- Thompson sampling implements probability matching

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

1. Use Bayes law to compute posterior distribution $p[\mathcal{R} \mid h_t]$
2. Sample a reward distribution $\mathcal{R}$ from posterior
3. Compute action-value function $Q(a) = \mathbb{E}[\mathcal{R}^a]$
4. Select action maximising value on sample, $a_t = \arg\max_{a \in \mathcal{A}} Q(a)$

Thompson sampling achieves Lai and Robbins lower bound!

Exploration/Exploitation Principles to MDPs

- The same principles for exploration/exploitation apply to MDPs
  - **Naive Exploration**
  - Optimistic Initialisation
  - Optimism in the Face of Uncertainty
  - Probability Matching

Lecture Outline

1. Exploration Theory
2. **Course Logistics**
3. Exploration in Deep RL

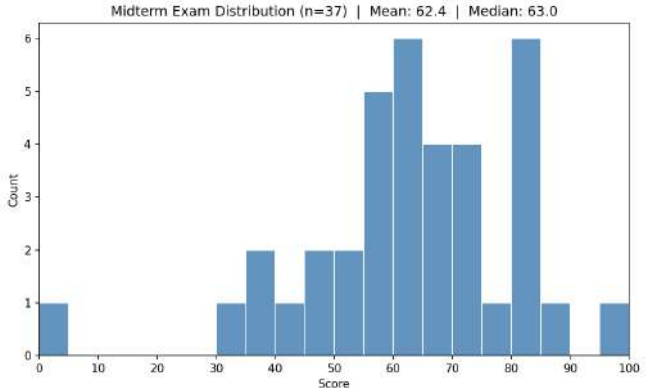
Course Logistics: Project Proposal

- The due date for project proposal is **Feb 27, 2026**.
- Please don't ask for extension. Penealty reduced to 5% for late submission per day
- Make sure you add a section for individual contribution.
- It is fine to say no contribution, incase your partner did not contribute.
- Problem deciding the project proposal? Write on Piazza/office hours.

Course Logistics: Lab 4

- Lab 4 tomorrow
- Due to very few turnout for previous labs, the lab notebook will be password protected.
- You will receive the password only in the tutorial.

Course Logistics: Midterm 1



Marking issue? Come to office hours today.

Course Logistics: Midterm 2

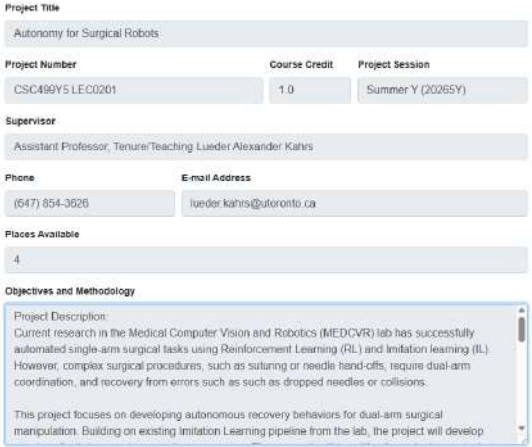
- Can not accommodate A2 peer review, due to TA load.
- Hence, we will have another quiz. (10%) !!
- It will be Multiple Choice Questions.
- Date: March 19th, 2026. syllbus: lecture 5-9

Course Logistics: Guest speaker

- Guest speaker: Dr. Mehdi Saeedi, from AMD.
- Topic: RL for gaming.
- Date: March 11th, 2026.

Summer Research Opportunity

- Our group has 6-8 summer research opportunities.
- Encourage you to apply using the ROP portal.



Project Title

Autonomy for Surgical Robots

Project Number	Course Credit	Project Session
CSC499Y5 LEC0201	1.0	Summer Y (20265Y)

Supervisor

Assistant Professor, Tenure/Teaching Lueder Alexander Kahrs

Phone	E-mail Address
(647) 854-3526	lueder.kahrs@utoronto.ca

Places Available

4

Objectives and Methodology

Project Description:
Current research in the Medical Computer Vision and Robotics (MEDCVR) lab has successfully automated single-arm surgical tasks using Reinforcement Learning (RL) and Imitation learning (IL). However, complex surgical procedures, such as suturing or needle hand-offs, require dual-arm coordination, and recovery from errors such as such as dropped needles or collisions.

This project focuses on developing autonomous recovery behaviors for dual-arm surgical manipulation. Building on existing Imitation Learning pipeline from the lab, the project will develop

Break

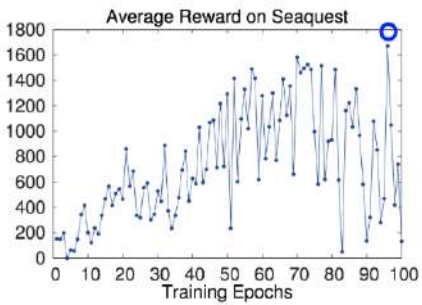
Lecture Outline

1. Exploration Theory
2. Course Logistics
3. **Exploration in Deep RL**

Exploration in Deep RL: Outline

Why Talking About Exploration-Exploitation?

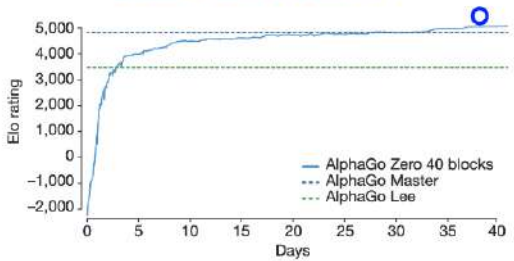
Superhuman performance



Mnih et al. [2015]

10 million frames

Beating world champion



Silver et al. [2016]

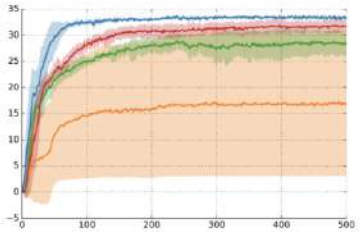
4.9 million games

Even best RL algorithms are very **sample inefficient**

Why do we need exploration?

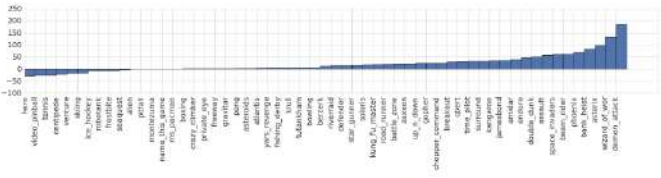
Better exploration may significantly **improve the sample efficiency**

*Optimism in face of uncertainty



Tang et al. [2017]

*Thompson sampling



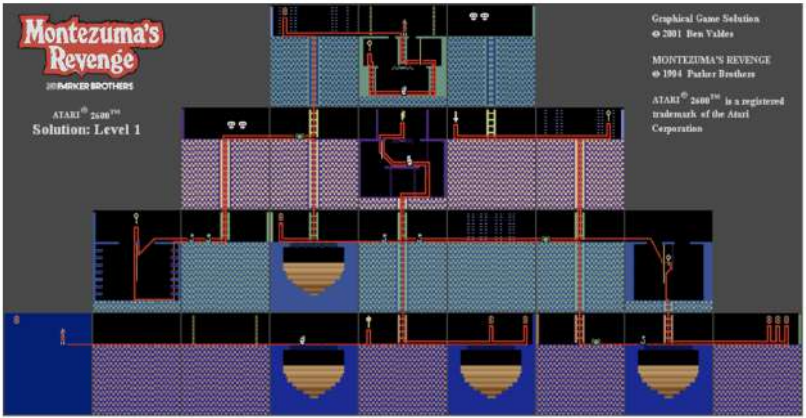
Fortunato et al. [2018]

All these methods use function approximation (e.g., deepNN)

these are easy

this is hard, *almost* impossible

Montezuma's Revenge: Level 1



The Four Ingredients Recipe

1. Build accurate estimators
2. Evaluate the uncertainty of the prediction
3. Define a mechanism to combine estimation and uncertainty
4. Execute the best policy

The Four Ingredients Recipe

Optimism in face of uncertainty

1. Build accurate estimators $\widehat{M}_k \Rightarrow V^{\pi}_{\widehat{M}_k}$
2. Evaluate the uncertainty of the estimators

$$\mathcal{B}^r_{hk}(s,a) := \left[\hat{r}_{hk}(s,a) - \beta^r_{hk}(s,a),\ \hat{r}_{hk}(s,a) + \beta^r_{hk}(s,a)\right]$$
$$\mathcal{B}^p_{hk}(s,a) := \left\{p(\cdot|s,a) \in \Delta(\mathcal{S}) : \|p - \hat{p}_{hk}\|_1 \leq \beta^p_{hk}(s,a)\right\}$$

3. Define a mechanism to combine estimation and uncertainty

$$Q_{hk}(s,a) = \hat{r}_{hk}(s,a) + b_{hk}(s,a) + \hat{p}_{hk}(s,a)\, V_{h+1,k}$$

"Practical" Limitations

Optimism in face of uncertainty

- Confidence intervals:

$$\beta_t^r(s,a) \propto \sqrt{\frac{\log(N_t(s,a)/\delta)}{N_t(s,a)}}, \qquad \beta_t^p(s,a) \propto \sqrt{\frac{S\log(N_t(s,a)/\delta)}{N_t(s,a)}}$$

- Solving:

$$\pi_t = \arg\max_{\pi} \max_{M \in \mathcal{M}_t} V_M^{\pi}$$

Exploration in Deep RL: Outline

Classification

1. **Count-based bonus**
2. **Prediction-based bonus**
3. **Bonus based on Auxiliary Task**

Count-based Exploration

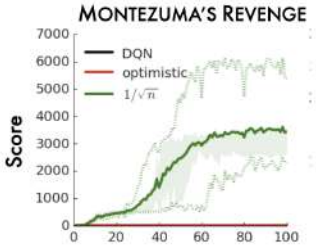
Count-based Exploration: General Scheme

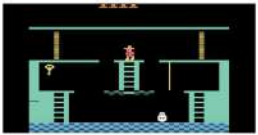
1. Estimate a "proxy" for the number of visits $\widetilde{N}(s_t)$
2. Add an exploration bonus to the rewards

$$\tilde{r}_t^+ = r_t + \beta_t \sqrt{\frac{1}{\widetilde{N}(s_t)}}$$

3. Run any DeepRL algorithm on $\mathcal{D}_t = (s_i, a_i, \tilde{r}_i^+, s_{i+1})$
4. $r_t^e \approx \sqrt{1/\widetilde{N}(s_t)}$ is inspired by theory (recall UCB)

Does It Work?





[Bellemare et al., *Unifying Count-Based Exploration and Intrinsic Motivation*, NeurIPS 2016.]

Count by Density Estimation

- Density estimation over a countable set $\mathcal{X}$ (i.e., observation space)

$$\rho_n(x) = \rho(x \mid x_1, \ldots, x_n) \approx P(X_{n+1} = x \mid x_1, \ldots, x_n)$$

- **Recording probability**

$$\rho'_n(x) = \rho(x \mid x_1, \ldots, x_n, x) \approx P(X_{n+2} = x \mid x_1, \ldots, x_n, X_{n+1} = x)$$

- **Pseudo-count** $\widetilde{N}_n(x)$ to imitate empirical counts s.t.

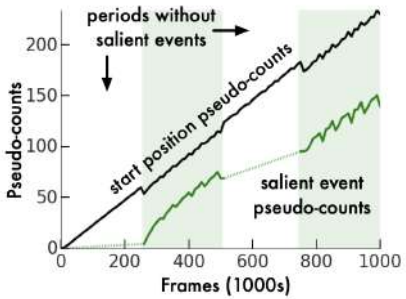
$$\frac{\widetilde{N}_n(x)}{\tilde{n}} = \rho_n(x) \leq \rho'_n(x) = \frac{\widetilde{N}_n(x) + 1}{\tilde{n} + 1}$$

$$\implies \widetilde{N}_n(x) = \frac{\rho_n(x)(1 - \rho'_n(x))}{\rho'_n(x) - \rho_n(x)} = \tilde{n}\,\rho_n(x)$$

[Bellemare et al., *Unifying Count-Based Exploration and Intrinsic Motivation*, NeurIPS 2016.
Ostrovski et al., *Count-Based Exploration with Neural Density Models*, ICML 2017.]

Count-based Exploration

Montezuma!



☞ Any density estimation algorithm (accurate for images)
e.g., GMM or CTS or PixelCNN

[Demo video]

[Bellemare et al., *Unifying Count-Based Exploration and Intrinsic Motivation*, NeurIPS 2016.]

Count-based Exploration

Algorithm 1: Count-based exploration through static hashing, using SimHash

1. Define state preprocessor $g : \mathcal{S} \to \mathbb{R}^D$
2. (In case of SimHash) Initialize $A \in \mathbb{R}^{k \times D}$ with entries drawn i.i.d. from the standard Gaussian distribution $\mathcal{N}(0, 1)$
3. Initialize a hash table with values $n(\cdot) \equiv 0$
4. **for** each iteration j **do**
5. Collect a set of state-action samples $\{(s_m, a_m)\}_{m=0}^{M}$ with policy π
6. Compute hash codes through any LSH method, e.g., for SimHash, $\phi(s_m) = \text{sgn}(Ag(s_m))$
7. Update the hash table counts $\forall m : 0 \le m \le M$ as $n(\phi(s_m)) \leftarrow n(\phi(s_m)) + 1$
8. Update the policy π using rewards $\left\{ r(s_m, a_m) + \frac{\beta}{\sqrt{n(\phi(s_m))}} \right\}_{m=0}^{M}$ with any RL algorithm

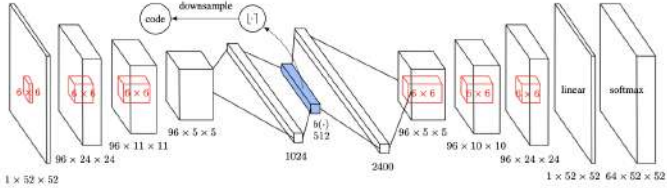
- Use **locality-sensitive hashing** to discretize the input
  - Encode the state into a k-dim vector by random projection small k = more hash collisions
  - Use the sign to discretize $\phi(s) \in \{-1, 1\}^k$
- *Count on discrete hashed-states*

✗ Difficult to define a good hashing function

[Tang et al., *#Exploration: A Study of Count-Based Exploration for Deep RL*, NeurIPS 2017.]

Count-based Exploration

- Improve counts by **learning a compression**

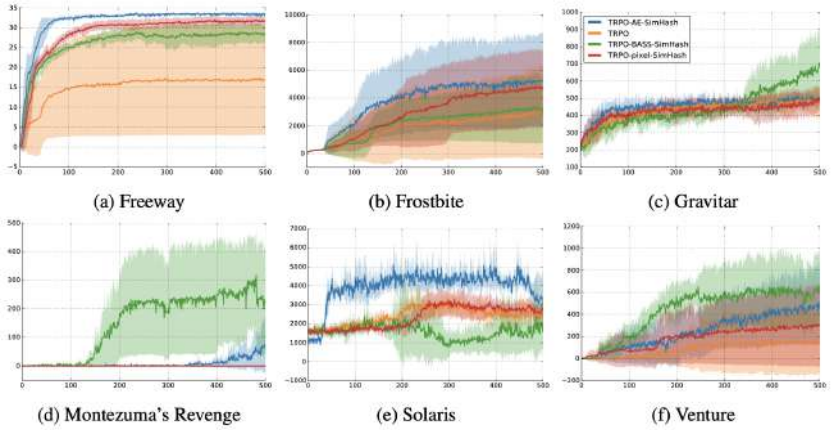


$$L\left(\{s_n\}_{n=1}^N\right) = -\frac{1}{N}\sum_{n=1}^{N}\left[\log p(s_n) - \frac{\lambda}{K}\sum_{i=1}^{D}\min\left\{(1-b_i(s_n))^2,\ b_i(s_n)^2\right\}\right]$$

- Entropy loss for the auto-encoder
- "Binarization" loss for the "projection"
- Use all past history to update the AE
- AE should not be updated too often. We need **stable codes!**

[Tang et al., *#Exploration: A Study of Count-Based Exploration for Deep RL*, NeurIPS 2017.]

Count-based Exploration



[Tang et al., *#Exploration: A Study of Count-Based Exploration for Deep RL*, NeurIPS 2017.]

Prediction-based Exploration

Forward Dynamics Prediction

- Given an encoding $\phi(s)$, learn a prediction model $f : (\phi(s_t), a_t) \mapsto \phi(s_{t+1})$
- Use the **prediction error** $e_t = \|\phi(s_{t+1}) - f(\phi(s_t), a_t)\|_2^2$ as exploration bonus $r_t^i \propto e_t$

How to learn $\phi(s)$?

- Pretrain the encoding (e.g., autoencoder)
- Learn it online using early samples

✗ difficult to predict every possible change in the transitions
○ exploration and representation are intertwined!

*the bonus is a normalized and scaled error

[Stadie et al., *Incentivizing Exploration in RL with Deep Predictive Models*, 2015.]

Is everything relevant?

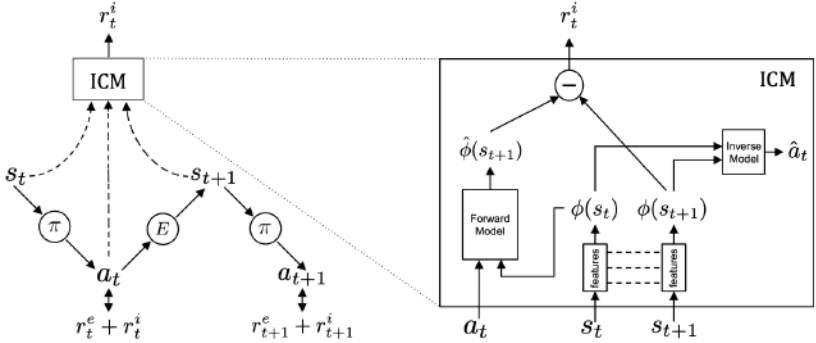
- **Idea:** [Pathak et al., 2017]
  predict only changes that depend on agent's actions, ignore the rest!
- **Mapping: representation learning problem**
  - learn embedding ϕ where only the information relevant to the action performed by the agent is represented (**controllability**)

Intrinsic Curiosity Module

Inverse dynamics: $h : (\phi(s_t), \phi(s_{t+1})) \mapsto \hat{a}_t$

Forward dynamics: $f : (\phi(s_t), a_t) \mapsto \hat{\phi}(s_{t+1})$

Intrinsic reward:
$r_t^i = \|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|_2^2$

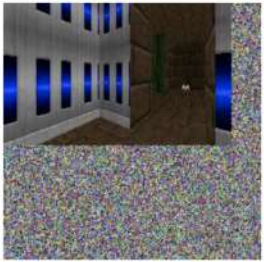


- Training: end-to-end training with auxiliary losses

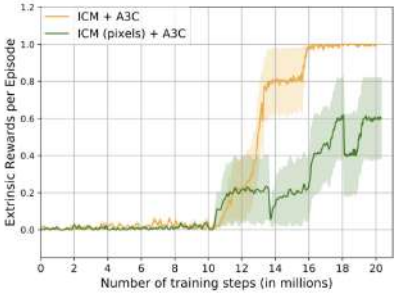
[Pathak et al., *Curiosity-Driven Exploration by Self-Supervised Prediction*, ICML 2017.]

Intrinsic Curiosity Module

- Intuition: inverse model h should be robust to uncontrollable components



(b) Input w/ noise



- *ICM (pixel) uses only forward dynamics
- **Inverse dynamics learning is at the base of many subsequent approaches** [Demo video]

[Pathak et al., *Curiosity-Driven Exploration by Self-Supervised Prediction*, ICML 2017.]

Study of Curiosity Driven Exploration

- Mostly pure exploration problems with surprise-based reward

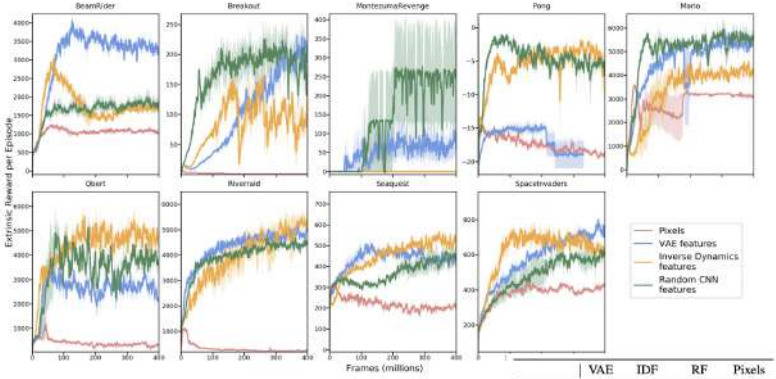
$$r_t = r_t^i = \|f(s_t, a_t) - \phi(s_{t+1})\|_2^2 \approx -\log p(s_{t+1} \mid s_t, a_t)$$

- Authors identified 3 properties of good representations: **Compact, Sufficient, Stable**
- Compared the following methods
  - **Pixel input:** $\phi(x) = x$
  - **Random features (RF)**
  - **Variational Autoencoders (VAE):** probabilistic encoder
  - **Inverse dynamic features (IDF):** as ICM

*experiments done in infinite horizon setting to avoid termination leaking information

[Burda et al., *Large-Scale Study of Curiosity-Driven Learning*, ICLR 2019.]

Results

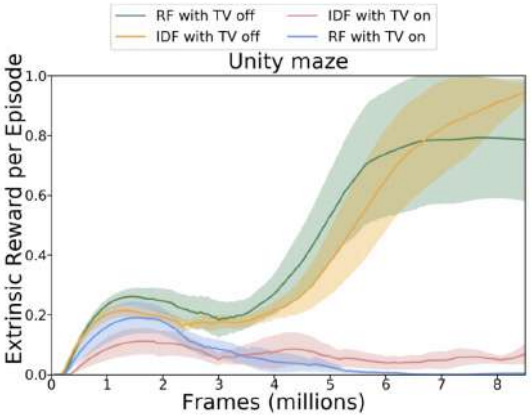


	VAE	IDF	RF	Pixels
Stable	No	No	Yes	Yes
Compact	Yes	Yes	Maybe	No
Sufficient	Yes	Maybe	Maybe	Yes

RF works quite well!

[Burda et al., *Large-Scale Study of Curiosity-Driven Learning*, ICLR 2019.]

Results: noisy TV



[Burda et al., *Large-Scale Study of Curiosity-Driven Learning*, ICLR 2019.]

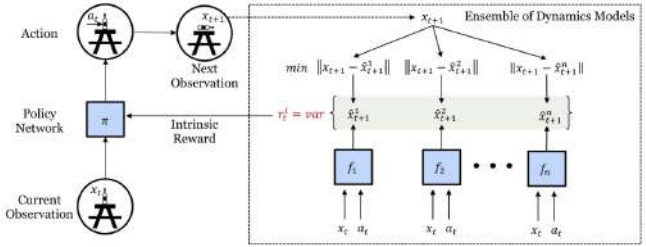
From One Model to Many

- All the methods used a **single model** to predict forward or inverse dynamics.
- We can also use **multiple models** and leverage **disagreement**:

$$\text{high disagreement} \implies \text{low confidence} \implies \text{need more data (exploration)}$$

Self-Supervised Exploration via Disagreement

- Ensemble method using multiple forward models (K models)



- **Intrinsic reward:** $r_t^i = \mathbb{E}_k \left[\| f_k(x_t, a_t) - \mathbb{E}_k[f_k(x_t, a_t)] \|_2^2 \right]$

☞ differentiable intrinsic reward

☞ can be paired with representation learning

✗ limitations of forward model learning

[Pathak et al., *Self-Supervised Exploration via Disagreement*, ICML 2019.]

Auxiliary Task

Exploration and Predictions

- So far, exploration bonus was based on
  - Generalized counts
  - Prediction error about dynamics
- But we can use **other predictions** for exploration $\Rightarrow$ value predictions

Random Network Distillation (RND)

- Randomly initialize two instances of the same NN (target θ^* and prediction θ_0)

$$f_{\theta^*} : \mathcal{S} \to \mathbb{R}^d, \qquad f_\theta : \mathcal{S} \to \mathbb{R}^d$$

- Train the prediction network minimizing loss w.r.t. the target network

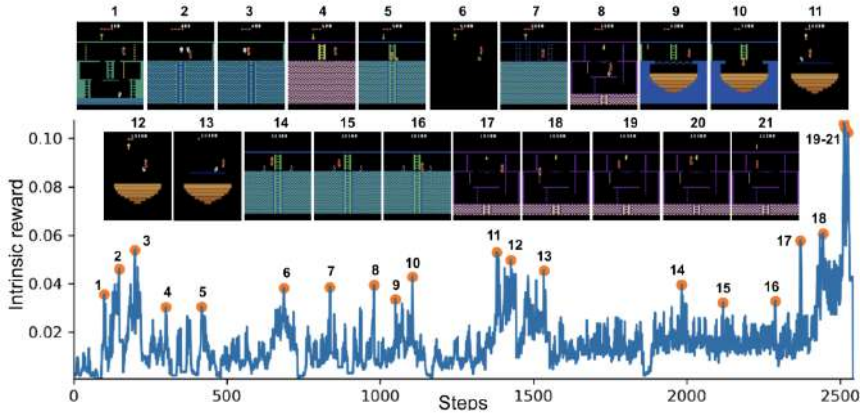
$$\theta_n = \arg\min_\theta \sum_{t=1}^{n} \|f_\theta(s_t) - f_{\theta^*}(s_t)\|^2$$

- Build "intrinsic" reward

$$r_t^i = \|f_\theta(s_t) - f_{\theta^*}(s_t)\|$$

- No model misspecification (f_θ can exactly predict f_{θ^*})
- Influence of learning dynamics can be reduced

Random Network Distillation (RND)



[Burda et al., *Exploration by Random Network Distillation*, ICLR 2019.]

Random Network Distillation (RND)

General architecture

- Separate extrinsic r_t^E and intrinsic reward r_t^I
- PPO (or any other approach) with two heads to estimate V^I and V^E
- Greedy policy w.r.t. $V^I + c\,V^E$

"Tricks"

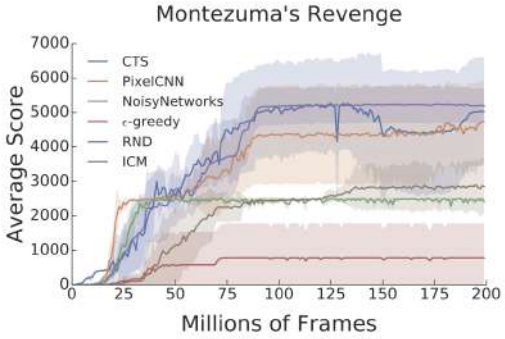
- Rewards should be in the same range
- Use different discount factors for intrinsic and extrinsic rewards
- Non-episodic setting results in better exploration

[Burda et al., *Exploration by Random Network Distillation*, ICLR 2019.]

Random Network Distillation (RND)

Montezuma!

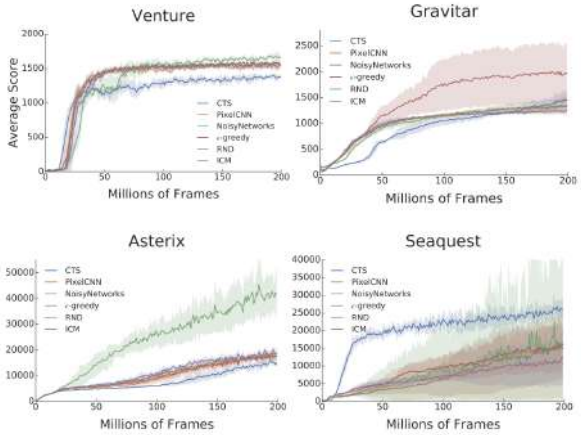
finds 22 out of the 24 rooms on the first level



[Demo video]

[Burda et al., *Exploration by Random Network Distillation*, ICLR 2019.]

Comparison: not all problems require same amount of exploration



[Taïga et al., *Benchmarking Bonus-Based Exploration Methods on the Arcade Learning Environment*, 2019.]

Exploration in Deep RL: Outline

Exploration in Deep RL: Outline

Exploration in Deep RL

- Several different techniques (we have seen only a small part)
- **No general solution**
- Exploration needs to account for **uncertainty in the predictions**
- Should account for **long-term effect**
- Exploration at the level of (value/policy/model) **parameters**

What is not covered here?

- Exploration and representation learning
- Unsupervised exploration
- Hierarchical exploration
- Parallel exploration
- Multi-task exploration
- Meta learning

Exploration Problems

- **Deterministic vs. stochastic**
- **Stable vs. unstable**
- **Deep vs. shallow**
- **With vs. without** offline data
- **Single vs. multiple** environments

Recommended Reading

- Bellemare et al. *Unifying Count-Based Exploration and Intrinsic Motivation.* NeurIPS 2016.
- Tang et al. *#Exploration: A Study of Count-Based Exploration for Deep RL.* NeurIPS 2017.
- Pathak et al. *Curiosity-Driven Exploration by Self-Supervised Prediction.* ICML 2017.
- Burda et al. *Large-Scale Study of Curiosity-Driven Learning.* ICLR 2019.
- Burda et al. *Exploration by Random Network Distillation.* ICLR 2019.
- Pathak et al. *Self-Supervised Exploration via Disagreement.* ICML 2019.
- Osband et al. *Deep Exploration via Bootstrapped DQN.* NeurIPS 2016.
- Fortunato et al. *Noisy Networks for Exploration.* ICLR 2018.
- Osband et al. *Randomized Prior Functions for Deep RL.* NeurIPS 2018.

Thank You!

Randomized Exploration: General Scheme inspired by Thompson Sampling

1. Estimate the parameters θ for either policy or value function
2. Add randomness to the parameters $\tilde{\theta} = \theta + \text{noise}$
3. Run the corresponding (greedy) policy

- **Remark:** changing weights induces a consistent, and potentially very complex, state-dependent change in policy over multiple time steps
  - $\Rightarrow$ long-term exploration
  - $\Rightarrow$ no dithering
- The randomness needs to represent "uncertainty"

Exploration via Randomization

- **Perturb observed rewards**
  - store samples $(s, a, s', r + \text{noise})$ run an RL algorithm on the perturbed data
- **Perturb parameters** (e.g., based on posterior uncertainty)
  - leverage uncertainty on the prediction
- **Randomized Value Function (RVF)**

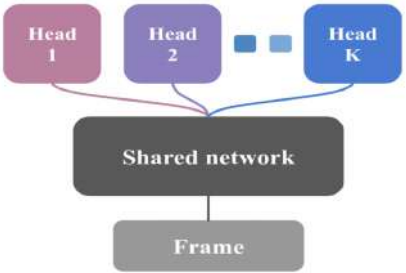
[Osband et al., *Deep Exploration via Bootstrapped DQN*, NeurIPS 2016.
Osband et al., *Randomized Prior Functions for Deep RL*, NeurIPS 2018.
Azizzadenesheli et al., *Efficient Exploration Through Bayesian Deep Q-Networks*, ITA 2018.
Lipton et al., *BBQ-Networks: Efficient Exploration in Deep Reinforcement Learning*, 2018.

Touati et al., *Randomized Value Functions via Multiplicative Normalizing Flows*, UAI 2019.]

Bootstrap DQN

DQN + bootstrapping $\approx$ Thompson sampling

- Define multiple value functions Q_k
- Update functions with different datasets
- Share part of the architecture
- another way of approximating a sample from posterior



[Osband et al., *Deep Exploration via Bootstrapped DQN*, NeurIPS 2016.]

Bootstrap DQN

- M_t determines the type of bootstrapping strategy

$$g_t^k = m_t^k \big(y_t^Q - Q_k(s_t, a_t; \theta)\big) \nabla_\theta Q_k(s_t, a_t; \theta)$$

with target $y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-)$

Algorithm 1 Bootstrapped DQN

Input: Value function networks Q with K outputs $\{Q_k\}_{k=1}^K$. Masking distribution M.
Let B be a replay buffer storing experience for training.
for each episode **do**
  Obtain initial state from environment s_0
  Pick a value function to act using $k \sim \text{Uniform}\{1, \ldots, K\}$
  for step $t = 1, \ldots$ until end of episode **do**
    Pick an action according to $a_t \in \arg\max_a Q_k(s_t, a)$
    Receive state s_{t+1} and reward r_t from environment, having taking action a_t
    Sample bootstrap mask $m_t \sim M$
    Add $(s_t, a_t, r_{t+1}, s_{t+1}, m_t)$ to replay buffer B
  end for
end for

[Osband et al., *Deep Exploration via Bootstrapped DQN*, NeurIPS 2016.]

Noisy Networks

- Normal NN layer $y = wx + b$
- Double the parameters with mean and variance

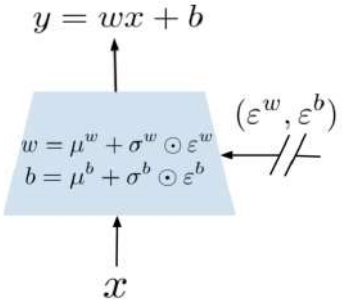
$$w \to \mu_w, \sigma_w \qquad \text{and} \qquad b \to \mu_b, \sigma_b$$

- Whenever a layer is evaluated draw $\varepsilon_w, \varepsilon_b \sim \mathcal{D}$
- Evaluate the "random" layer as

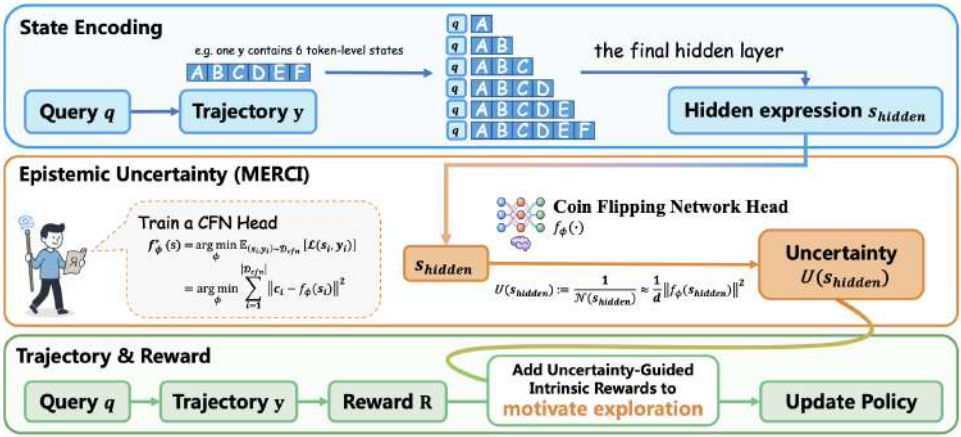
$$y = (\mu_w + \sigma_w \odot \varepsilon_w)\, x + \mu_b + \sigma_b \odot \varepsilon_b$$

- Let $\zeta = (\mu_w, \sigma_w, \mu_b, \sigma_b)$, define the expected loss $L(\zeta) = \mathbb{E}_\varepsilon\big[L(\zeta, \varepsilon)\big]$
- Gradient estimation update

[Fortunato et al., *Noisy Networks for Exploration*, ICLR 2018.]

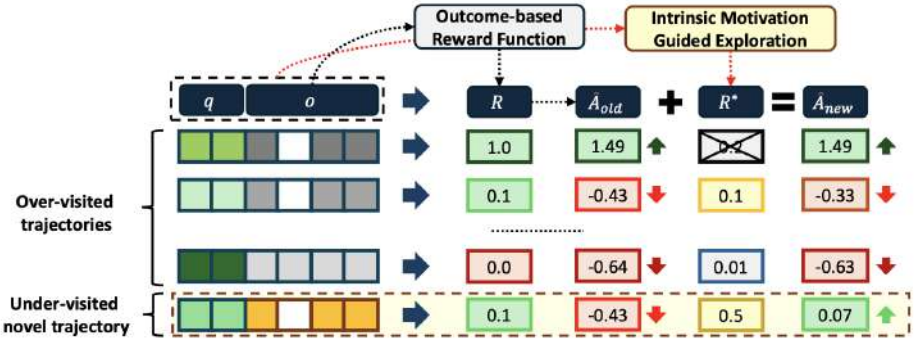


Exploration for LLM: State-of-the-art



[Chang et al., *Count Counts: Motivating Exploration in LLM Reasoning with Count-Based Intrinsic Reward*, ICLR 2026.]

Exploration for LLM



[Gao et al., *Navigate the Unknown: Enhancing LLM Reasoning with Intrinsic Motivation Guided Exploration*, arXiv.]