Exploration in Deep Reinforcement Learning

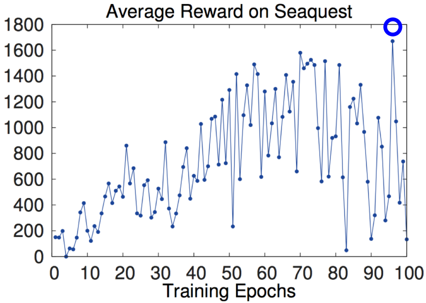
Matteo Pirotta
Facebook AI Research

ANITI's Reinforcement Learning Virtual School (RLVS-ANITI)
April 2, 2021

FACEBOOK AI

Why Talking About Exploration-Exploitation?

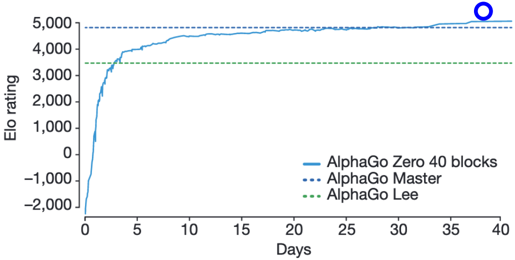
Superhuman performance



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**

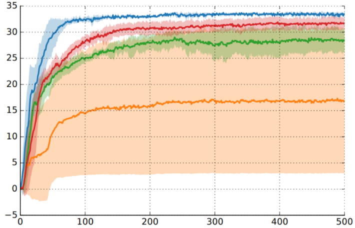
Efficiency

- Sample efficiency
- Computational efficiency

Why do we need exploration?

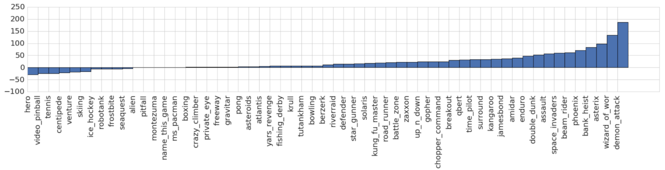
Better exploration may significantly **improve the sample efficiency**

**Optimism in face of uncertainty*



Tang et al. [2017]

**Thompson sampling*



Fortunato et al. [2018]

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

Exploration in Deep RL: Outline

These slides and additional material on my website and

https://rlgammazero.github.io/

Super-fast intro to MDPs

Only for notation

Markov decision process (MDP) is a tuple $M = \langle \mathcal{S}, \mathcal{A}, r, p \rangle$

- State space $\mathcal{S}$
- Action space $\mathcal{A}$
- Transition function $p(\cdot|s,a) \in \Delta(\mathcal{S})$
- Reward distribution with expectation $r(s,a)$

Policy: $\pi : \mathcal{S} \to \Delta(\mathcal{A})$

Value functions:

$$Q^\pi(s,a) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t | s_0 = s, a_0 = a\right]$$
$$V^\pi(s) = \mathbb{E}_{a\sim\pi(s)}[Q^\pi(s,a)]$$

Optimal policy: $\pi^\star = \arg\max_\pi \{V^\pi\}$

Online Learning Problem

Input: $\mathcal{S}$, $\mathcal{A}$ $\cancel{r_h, p_h}$

Initialize $Q_1(s,a) = 0$, $\mathcal{D}_1 = \emptyset$

for $k = 1, \dots, K$ **do** `// episodes`

  Define π_k based on Q_k

  for $h = 1, \dots, H$ **do**

    Execute $a_{hk} = \pi_k(s_{hk})$

    Observe r_{hk} and $s_{h+1,k}$

  end

  Add trajectory $(s_{hk}, a_{hk}, r_{hk})_{h \geq 1}$ to $\mathcal{D}_{k+1}$

  Compute Q_{k+1} from $\mathcal{D}_{k+1}$

end

What is Wrong with Q-learning with ϵ-greedy?

- ϵ-greedy strategy

$$a_k = \begin{cases} \arg\max_{a \in \mathcal{A}} \; Q_k(s_k, a) & \text{w.p.} \quad 1 - \epsilon_k, \\ \mathcal{U}(\mathcal{A}) & \text{otherwise.} \end{cases}$$

- Q-learning update

$$Q_{k+1}(s_k, a_k) = (1 - \alpha_k) Q_k(s_k, a_k) + \alpha_k \left(r_k + \max_{a' \in \mathcal{A}} Q_{h+1,k}(s_{k+1}, a') \right)$$

*$H = 1$

What is Wrong with Q-learning with ϵ-greedy?

- ϵ-greedy strategy

$$a_k = \begin{cases} \arg\max_{a \in \mathcal{A}} \; Q_k(s_k, a) & \text{w.p.} \quad 1 - \epsilon_k, \\ \mathcal{U}(\mathcal{A}) & \text{otherwise.} \end{cases}$$

- Q-learning update

$$Q_{k+1}(s_k, a_k) = (1 - \alpha_k) Q_k(s_k, a_k) + \alpha_k \left(r_k + \max_{a' \in \mathcal{A}} Q_{h+1,k}(s_{k+1}, a') \right)$$

The exploration strategy relies on **biased** estimates Q_k

*$H = 1$

What is Wrong with Q-learning with ϵ-greedy?

- ϵ-greedy strategy

$$a_k = \begin{cases} \arg\max_{a \in \mathcal{A}} Q_k(s_k, a) & \text{w.p. } 1 - \epsilon_k, \\ \mathcal{U}(\mathcal{A}) & \text{otherwise.} \end{cases}$$

- Q-learning update

$$Q_{k+1}(s_k, a_k) = (1 - \alpha_k) Q_k(s_k, a_k) + \alpha_k \Big(r_k + \max_{a' \in \mathcal{A}} Q_{h+1,k}(s_{k+1}, a') \Big)$$

The exploration strategy relies on **biased** estimates Q_k

Samples are used **once**

*$H = 1$

What is Wrong with Q-learning with ϵ-greedy?

- ϵ-greedy strategy

$$a_k = \begin{cases} \arg\max_{a \in \mathcal{A}} Q_k(s_k, a) & \text{w.p. } 1 - \epsilon_k, \\ \mathcal{U}(\mathcal{A}) & \text{otherwise.} \end{cases}$$

- Q-learning update

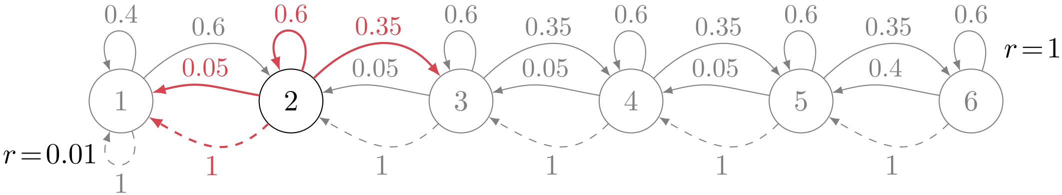
$$Q_{k+1}(s_k, a_k) = (1 - \alpha_k) Q_k(s_k, a_k) + \alpha_k \left(r_k + \max_{a' \in \mathcal{A}} Q_{h+1,k}(s_{k+1}, a') \right)$$

- The exploration strategy relies on **biased** estimates Q_k
- Samples are used **once**
- **Dithering effect:** exploration is not effective in covering the state space
- **Policy shift:** the policy changes at each step

$^*H = 1$

River Swim: Markov Decision Processes

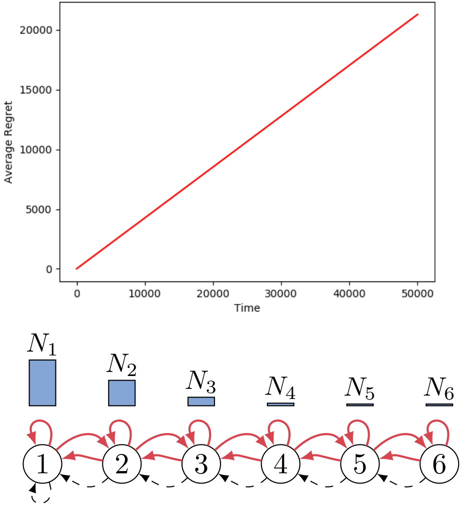
Strehl and Littman [2008]



- $\mathcal{S} = \{1, 2, 3, 4, 5, 6\}$, $\mathcal{A} = \{L, R\}$
- $\pi_L(s) = L$, $\pi_R(s) = R$

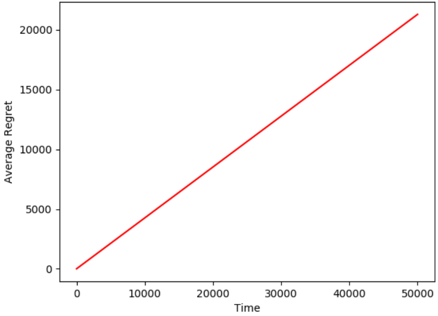
River Swim: Q-learning w\ ϵ-greedy Exploration

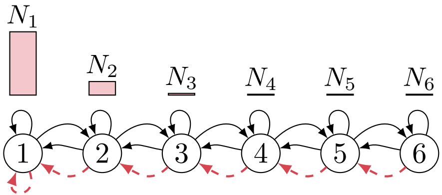
- $\epsilon_t = 1.0$



River Swim: Q-learning w\ ϵ-greedy Exploration

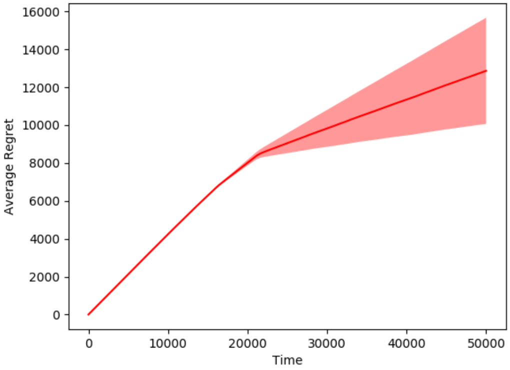
- $\epsilon_t = 1.0$
- $\epsilon_t = 0.5$





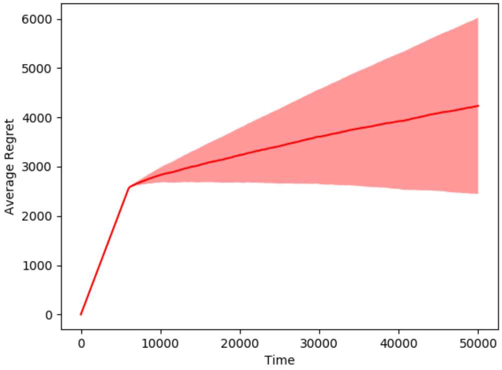
River Swim: Q-learning w\ ϵ-greedy Exploration

- $\epsilon_t = 1.0$
- $\epsilon_t = 0.5$
- $\epsilon_t = \dfrac{\epsilon_0}{(N(s_t) - 1000)^{2/3}}$



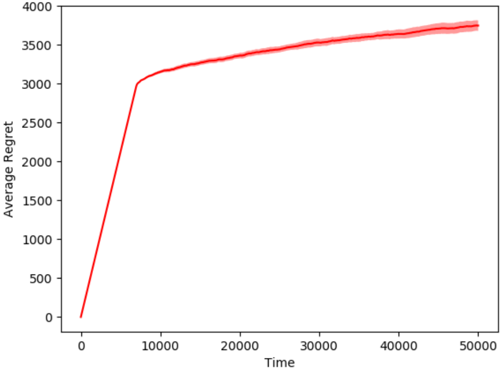
River Swim: Q-learning w\ ϵ-greedy Exploration

- $\epsilon_t = 1.0$
- $\epsilon_t = 0.5$
- $\epsilon_t = \dfrac{\epsilon_0}{(N(s_t) - 1000)^{2/3}}$
- $\epsilon_t = \begin{cases} 1.0 & t < 6000 \\ \dfrac{\epsilon_0}{N(s_t)^{1/2}} & \text{otherwise} \end{cases}$



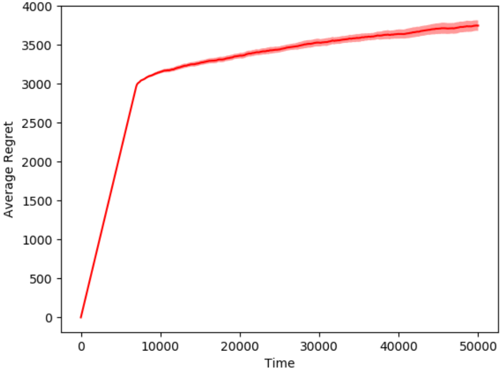
River Swim: Q-learning w\ ϵ-greedy Exploration

- $\epsilon_t = 1.0$
- $\epsilon_t = 0.5$
- $\epsilon_t = \dfrac{\epsilon_0}{(N(s_t) - 1000)^{2/3}}$
- $\epsilon_t = \begin{cases} 1.0 & t < 6000 \\ \dfrac{\epsilon_0}{N(s_t)^{1/2}} & \text{otherwise} \end{cases}$
- $\epsilon_t = \begin{cases} 1.0 & t < 7000 \\ \dfrac{\epsilon_0}{N(s_t)^{1/2}} & \text{otherwise} \end{cases}$



River Swim: Q-learning w\ ϵ-greedy Exploration

- $\epsilon_t = 1.0$
- $\epsilon_t = 0.5$
- $\epsilon_t = \dfrac{\epsilon_0}{(N(s_t) - 1000)^{2/3}}$
- $\epsilon_t = \begin{cases} 1.0 & t < 6000 \\ \dfrac{\epsilon_0}{N(s_t)^{1/2}} & \text{otherwise} \end{cases}$
- $\epsilon_t = \begin{cases} 1.0 & t < 7000 \\ \dfrac{\epsilon_0}{N(s_t)^{1/2}} & \text{otherwise} \end{cases}$



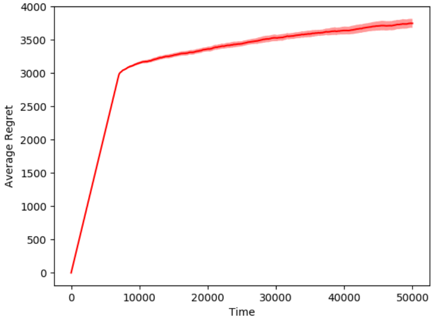
Tuning the ϵ schedule is **difficult and problem dependent**

River Swim: Q-learning w\ ϵ-greedy Exploration

Main drawbacks of Q-learning with ϵ-greedy

- ϵ-greedy performs *undirected* exploration
- *Inefficient use* of samples

Regret: $\Omega\left(\min\{T, A^{H/2}\}\right)$ [Jin et al., 2018]

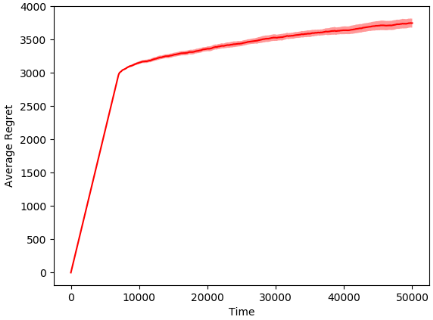


River Swim: Q-learning w\ ϵ-greedy Exploration

Main drawbacks of Q-learning with ϵ-greedy

- ϵ-greedy performs *undirected* exploration
- *Inefficient use* of samples

Regret: $\Omega\left(\min\{T, A^{H/2}\}\right)$ [Jin et al., 2018]



Uncertainty-driven exploration-exploitation

What do we know?

In *tabular MDPs* (finite state and actions), we have several approaches for exploration

[Jaksch et al., 2010, Zhang and Ji, 2019, Fruit et al., 2018b,a, 2020, Qian et al., 2019, Wei et al., 2020, Hao et al., 2021, Gong and Wang, 2020, Abb, 2019, Azar et al., 2017, Dann et al., 2017, Zanette and Brunskill, 2018, Jin et al., 2018, Zanette and Brunskill, 2019, Zhang et al., 2020, Menard et al., 2021, Neu and Pike-Burke, 2020, Efroni et al., 2019, Cai et al., 2020, Shani et al., 2020]

and we have *efficient optimal algorithm* (i.e., matching the statistical lower-bound)

Exploration in Deep RL: Outline

Exploration in Tabular MDPs

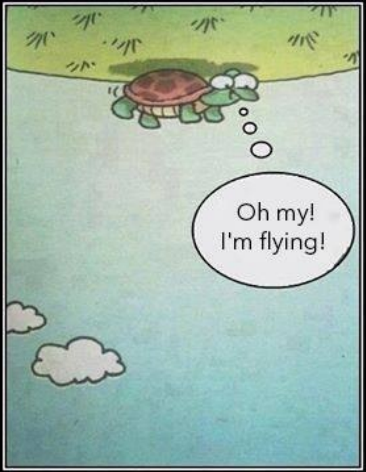
The Four Ingredients Recipe

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

Principles:

- Optimism in face of uncertainty (i.e., upper-confidence bounds)
- Thompson Sampling

The Optimism Principle: Intuition



OPTIMISM
It's the best way to see life.

(img from web)

Optimism

At each episode k, we must use an *estimate* Q_k such that

$$\forall (s,a), \qquad Q_k(s,a) \geq Q^\star(s,a) \quad (whp)$$

to compute the policy (since we *don't know* r *and* p):

$$a_{hk} = \arg\max_a Q_k(s_{hk}, a)$$

*$Q^\star(s,a) = \max_a \left\{ r(s,a) + \sum_{s'} p(s'|s,a) \max_{a'} Q^\star(s',a') \right\}$, p and r are unknown

Optimism: Model Optimism and Value Optimism

Optimism in model space

construct a confidence set around p and r and **jointly** optimize over models and policies

Optimism in value space

construct upper confidence bounds directly on the optimal value function $V^\star$

Both approaches lead to optimism $Q_k(s,a) \geq Q^\star(s,a)$

Optimism: *Model Optimism*

- Build *confidence set* around empirical transitions such that

$$D\big(p(\cdot|s,a), \widehat{p}_k(\cdot|s,a)\big) \le \beta_k^p(s,a)$$
$$|r(s,a) - \widehat{r}_k(s,a)| \le \beta_k^r(s,a)$$

and, with high probability

$$p(s,a) \in B_k^p(s,a), \quad r(s,a) \in B_k^r(s,a)$$

- Compute optimistic policy and model

$$(M_k, \pi_k) \in \arg\max_{M=(p,r)\in(B^p,B^r),\pi} \left\{ V_{1,M}^{\pi} \right\}$$

Example: [Jaksch et al., 2010]
Weissman inequality implies that $D = \|\cdot\|_1$
and $\beta_{hk}^p(s,a) \approx C\sqrt{S/N_k(s,a)}$
Hoeffding for reward leads to $\beta_k^r(s,a) \approx C\sqrt{1/N_k(s,a)}$

$N_k(s,a) = \#$ visits to (s,a) so far (before k)

$$\widehat{p}_k(s'|s,a) = \frac{N_k(s,a,s')}{N_k(s,a)}$$

$$\widehat{r}_k(s,a) = \frac{1}{N_k(s,a)} \sum_{t=1}^{k} r_t \cdot \delta_{sat}$$

Optimism: *Value Optimism*

- Compute exploration bonus $b_k(s,a)$
- Update estimated $Q^\star$
  - *Model-based*

    e.g., value iteration on $\overline{M}_k = (\mathcal{S}, \mathcal{A}, \widehat{r}_k + b_k, \widehat{p}_k)$
  - *Model-free*

    e.g., Q-learning update

$$Q_{k+1}(s_k, a_k) = (1-\alpha_k)Q_k(s_k, a_k) + \alpha_k \left(r_k + b_k + \max_{a' \in \mathcal{A}} Q_{h+1,k}(s_{k+1}, a') \right)$$

Example: [Azar et al., 2017]
$b_k(s,a) = C\sqrt{1/N_k(s,a)}$

$N_k(s,a) = \#$ visits to (s,a) so far (before k)

$\widehat{p}_k(s'|s,a) = \frac{N_k(s,a,s')}{N_k(s,a)}$

$\widehat{r}_k(s,a) = \frac{1}{N_k(s,a)} \sum_{t=1}^{k} r_t \cdot \delta_{sat}$

Thompson Sampling

- Keeps track of a *belief over models or Q-values*

$$\mathbb{P}(\theta|\mathcal{D}_{k-1})$$

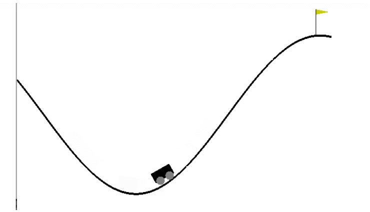
- *Samples* a plausible realization from the *posterior*

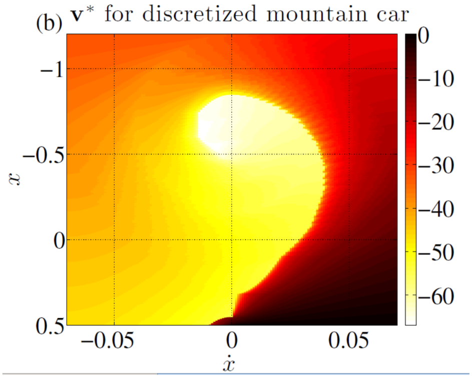
$$\theta_k \sim \mathbb{P}(\cdot|\mathcal{D})$$

- *Acts* with such realization (i.e., believes θ_k is the true value)

What happens if we move to general problems (i.e., non tabular)?

Example: Mountain Car





Function Approximation

Theory of exploration has focused on (with several structural assumptions)

- Linear function approximation
- Kernel approximation
- General function approximation
- Neural exploration

Optimism is still a key ingredient!

Still not very practical!

General Function Approximation

The agent is given a *function class*

$$\mathcal{F} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$$

to *approximate $Q^\star$*

<u>Idea:</u>

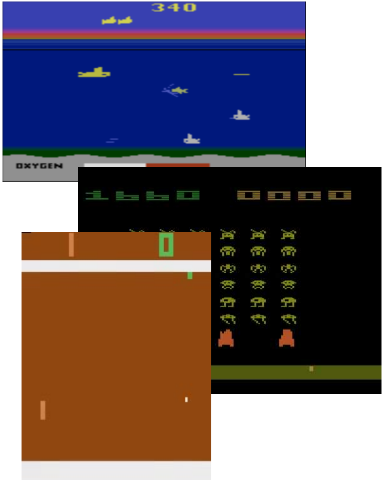
- Build confidence interval $\mathcal{B}$ of plausible $Q^\star$
- Optimistic planning, i.e., pick the best in the confidence set

⚠ *Extremely challenging without further assumptions!*
e.g., realizability and completeness

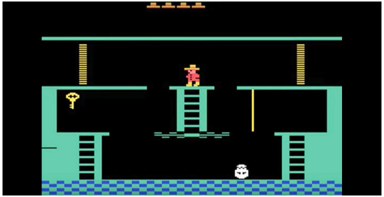
next practical algorithms!

Exploration in Deep RL: Outline

these are easy



this is hard, *almost* impossible

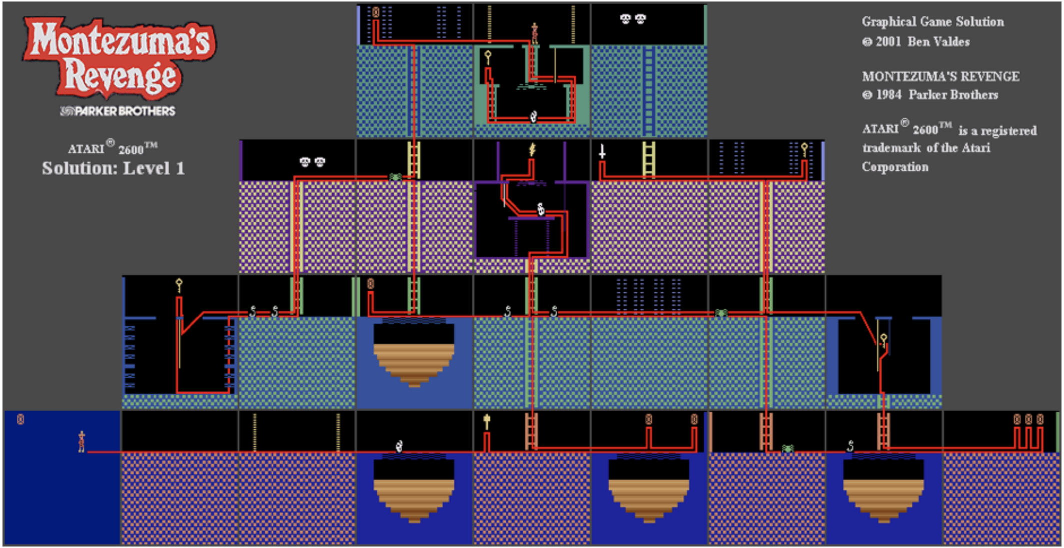


Why?

Random exploration sometimes work!
PONG GIF

Montezuma with random actions!
Link

Montezuma's Revenge: Level 1



Exploration Issues

1. Discovery
   Unknown State Space, Partial Observability, Sparse Reward
2. Controllability
   Predictability, Learnability
3. Representation Learning

. . . and probably more!

Exploration Issues

1. Discovery
   Unknown State Space, Partial Observability, Sparse Reward
2. *Controllability*
   Predictability, Learnability
3. Representation Learning

. . . and probably more!

Controllability

In front of a screen full of white noise conveying a lot of information and "novelty" and "surprise" in the traditional sense of Boltzmann and Shannon, however, it will experience highly unpredictable and fundamentally incompressible data – [Schmidhuber, 2010]

Controllability

> *In front of a screen full of white noise conveying a lot of information and "novelty" and "surprise" in the traditional sense of Boltzmann and Shannon, however, it will experience highly unpredictable and fundamentally incompressible data* – [Schmidhuber, 2010]

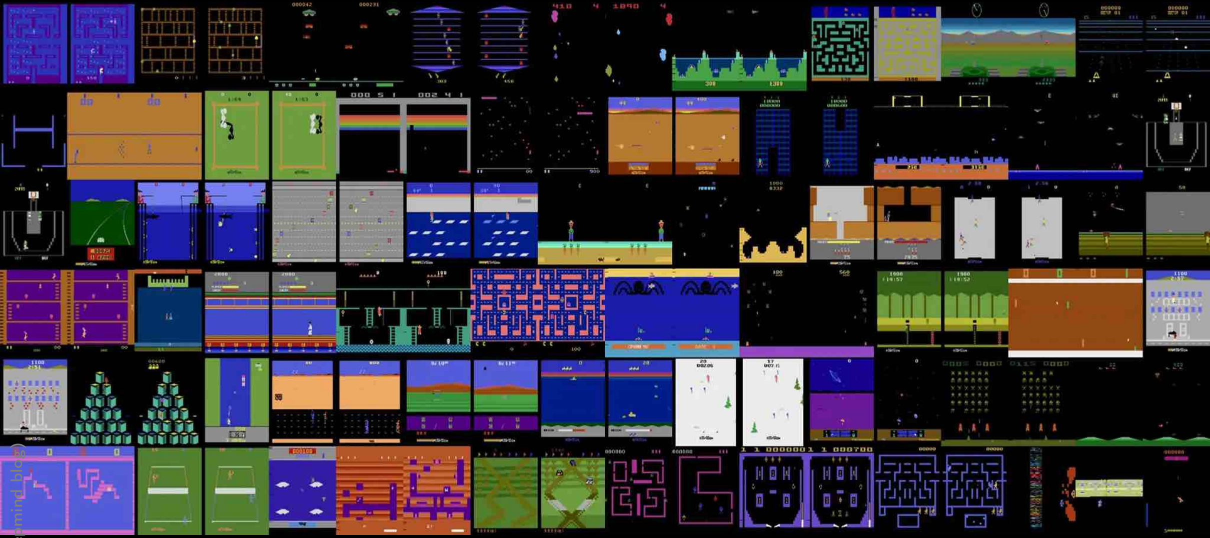
- States can be interesting due to an intrinsic variability
- Agent may get trapped by these states

video1 video2

Are these states relevant? *Probably not if they are uncontrollable and/or unpredictable*

(img from wikipedia)

Benchmark: Atari 57



(img for Deepmind blog)

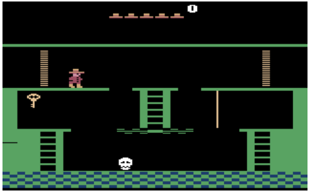
Equally difficult? *No*

1. Long-term credit assignment

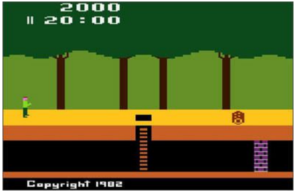
Skiing

Solaris

2. Exploration



Montezuma's Revenge



Pitfall

Exploration in Deep RL: Outline

Idea

① Augment the reward with an additional (vanishing) reward term

$$r_t = \underbrace{r_t^e}_{\text{extrinsic reward (standard)}} + \beta \underbrace{r_t^i}_{\text{intrinsic}}$$

- r^e: *extrinsic reward* (*task reward*)
- r^i: *intrinsic reward* (*exploration bonus*)

② Run any algorithm using the new reward r_t^+

Typical Objective

- Discover novel (or controllable) states

  encourage the agent to discover novel information

- Improve knowledge about the environment

  encourage the agent to perform actions to reduce uncertainty in predicting model evolution

- ...

Intrinsic reward is often inspired by psychology (*intrinsic motivation*), e.g., curiosity driven exploration (*self-supervised*) when $r_t^e = 0$

Arbitrary ☺ classification

- Count-based bonus
- Prediction-based bonus
- Bonus based on Auxiliary Task

Count-based Exploration

Count-based Exploration

General Scheme

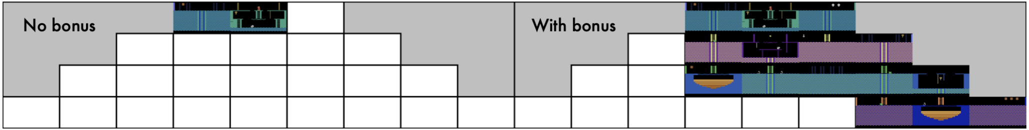
1. Estimate a "proxy" for the *number of visits* $\widetilde{N}(s_t)$
2. Add an *exploration bonus* to the rewards

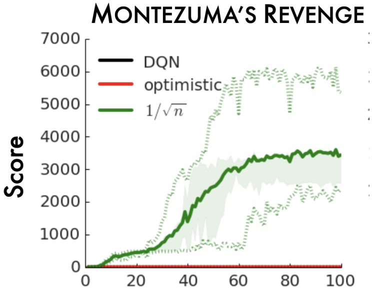
$$\widetilde{r}_t^+ = r_t + \beta_t \sqrt{\frac{1}{\widetilde{N}(s_t)}}$$

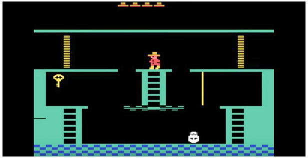
3. Run *any DeepRL* algorithm on $\mathcal{D}_t = \left\{(s_i, a_i, \widetilde{r}_i^+, s_{i+1})\right\}$

$r_t^e \approx \sqrt{1/\widetilde{N}(s_t)}$ is inspired by theory (recall UCB)

Does it work?







(img from [Bellemare et al., 2016])

Count by Density Estimation

[Bellemare et al., 2016, Ostrovski et al., 2017]

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

$$\rho_n(x) = \rho(x|x_1,\dots,x_n) \approx \mathbb{P}\big[X_{n+1} = x|x_1,\dots,x_n\big]$$

- Recording probability

$$\rho'_n(x) = \rho(x|x_1,\dots,x_n,x) \approx \mathbb{P}\big[X_{n+2} = x|x_1,\dots,x_n,X_{n+1} = x\big]$$

probability of x after observing a new occurrency of x

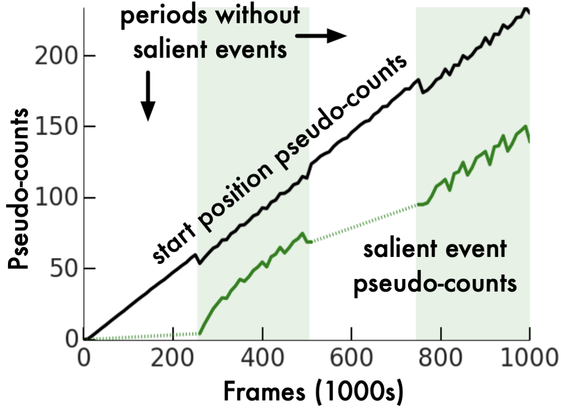
Pseudo count $\widetilde{N}_n(x)$ to imitate empirical count s.t.

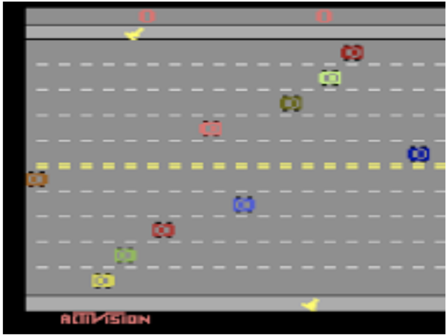
$$\frac{\widetilde{N}_n(x)}{\widetilde{n}} = \rho_n(x) \leq \rho'_n(x) = \frac{\widetilde{N}_n(x)+1}{\widetilde{n}+1}$$

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

Count-based Exploration

[Bellemare et al., 2016, Ostrovski et al., 2017]





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

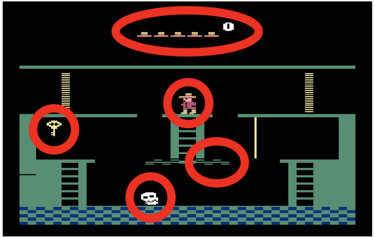
(img from [Bellemare et al., 2016])

Count-based Exploration

Bellemare et al. [2016], Ostrovski et al. [2017]

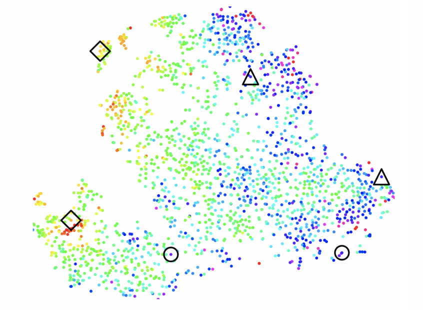
Montezuma!

What to Count?

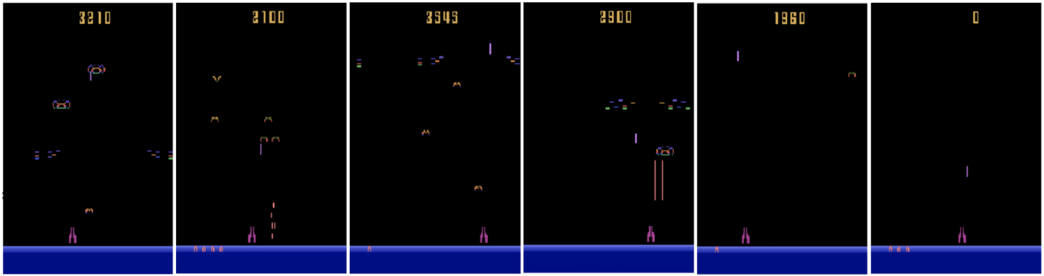


Representation Learning? *learn an embedding of state*

What is important to learn?

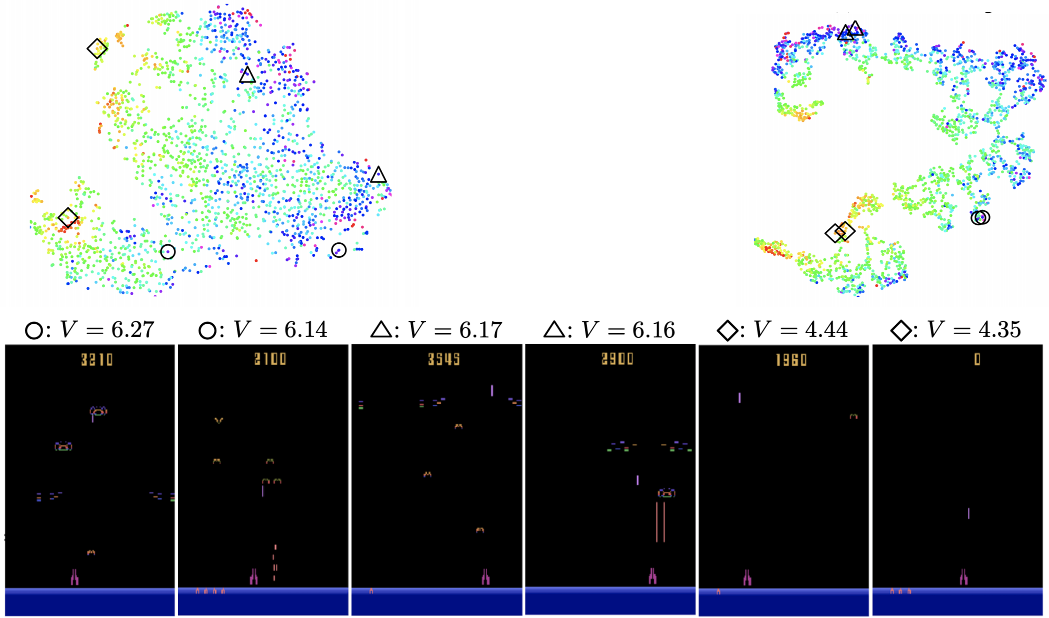


$\bigcirc$: $V = 6.27$ $\bigcirc$: $V = 6.14$ $\triangle$: $V = 6.17$ $\triangle$: $V = 6.16$ $\diamond$: $V = 4.44$ $\diamond$: $V = 4.35$



(img from [Sun et al., 2019])

What is important to learn?



(img from [Sun et al., 2019])

Count-based Exploration

[Tang et al., 2017]

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

Define state preprocessor $g : \mathcal{S} \to \mathbb{R}^D$
(In case of SimHash) Initialize $A \in \mathbb{R}^{k \times D}$ with entries drawn i.i.d. from the standard Gaussian distribution $\mathcal{N}(0, 1)$
Initialize a hash table with values $n(\cdot) \equiv 0$
for each iteration j **do**
  Collect a set of state-action samples $\{(s_m, a_m)\}_{m=0}^M$ with policy π
  Compute hash codes through any LSH method, e.g., for SimHash, $\phi(s_m) = \mathrm{sgn}(Ag(s_m))$
  Update the hash table counts $\forall m : 0 \le m \le M$ as $n(\phi(s_m)) \leftarrow n(\phi(s_m)) + 1$
  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 project
    small k = more hash collisions
  - Use the sign to discretize
    small $\phi(s) \in \{-1, 1\}^k$
- *Count on discrete hashed-states*

(img from [Tang et al., 2017])

Count-based Exploration

[Tang et al., 2017]

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

Define state preprocessor $g : \mathcal{S} \to \mathbb{R}^D$

(In case of SimHash) Initialize $A \in \mathbb{R}^{k \times D}$ with entries drawn i.i.d. from the standard Gaussian distribution $\mathcal{N}(0, 1)$

Initialize a hash table with values $n(\cdot) \equiv 0$

for each iteration j **do**

  Collect a set of state-action samples $\{(s_m, a_m)\}_{m=0}^{M}$ with policy π

  Compute hash codes through any LSH method, e.g., for SimHash, $\phi(s_m) = \mathrm{sgn}(Ag(s_m))$

  Update the hash table counts $\forall m : 0 \leq m \leq M$ as $n(\phi(s_m)) \leftarrow n(\phi(s_m)) + 1$

  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 project
    small k = more hash collisions
  - Use the sign to discretize
    small $\phi(s) \in \{-1, 1\}^k$
- *Count on discrete hashed-states*

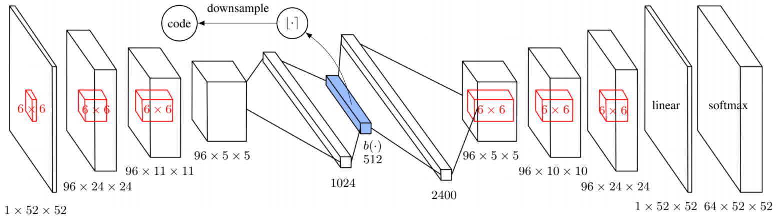
Difficult to define a good hashing function

(img from [Tang et al., 2017])

Count-based Exploration

[Tang et al., 2017]

Improve counts by learning a compression



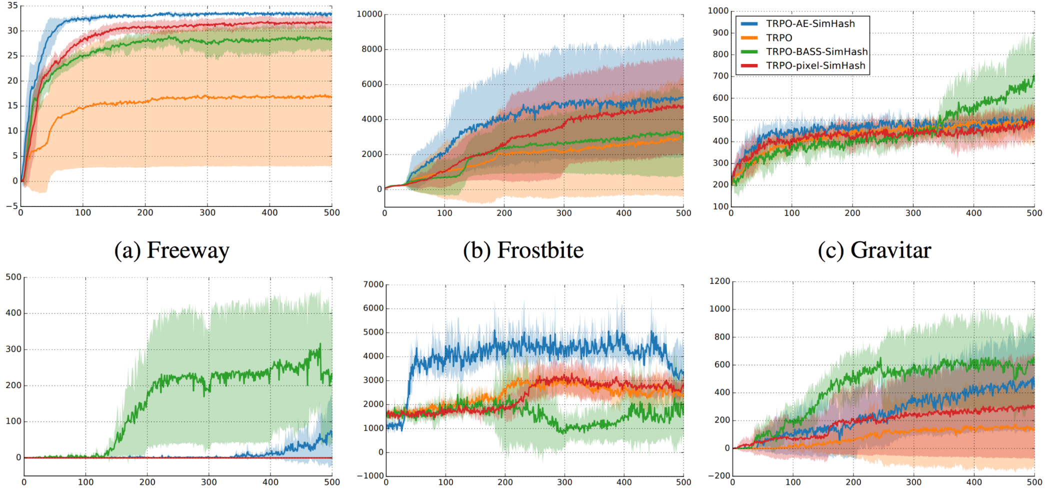
$$L\left(\{s_n\}_{n=1}^{N}\right) = -\frac{1}{N}\sum_{n=1}^{N}\left[\log p(s_n) - \frac{\lambda}{K}\sum_{i=1}^{D}\min\left\{(1-b_i(s_n))^2, b_i(s_n)^2\right\}\right]$$

- 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!**

(img from [Tang et al., 2017])

Count-based Exploration

[Tang et al., 2017]



(a) Freeway (b) Frostbite (c) Gravitar

(d) Montezuma's Revenge (e) Solaris (f) Venture

(img from [Tang et al., 2017])

Prediction-based Exploration

Forward Dynamics Prediction

[Stadie et al., 2015]

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

⚠ *exploration and representation are intertwined!*

difficult to predict every possible change in the transitions

*the bonus is a normalized and scaled error

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

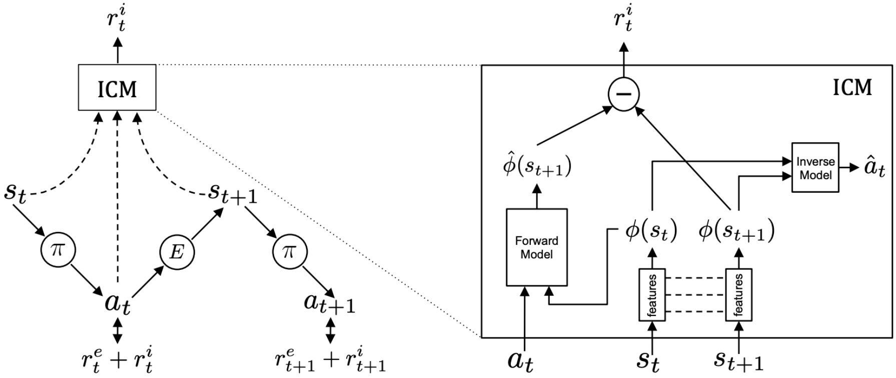
[Pathak et al., 2017]

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

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

Intrinsic reward:

$r_t^i = \|\widehat{\phi}(s_{t+1}) - \phi(s_{t+1})\|_2^2$

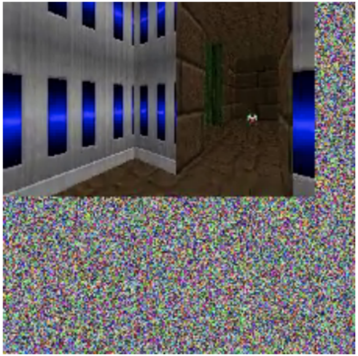


Training: end-to-end training with auxiliary losses

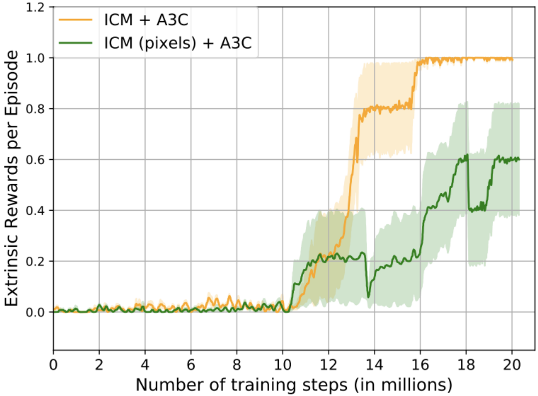
Intrinsic Curiosity Module

[Pathak et al., 2017]

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

Study of Curiosity Driven Exploration

[Burda et al., 2019a]

- Mostly pure exploration problems with *surprise*-based reward

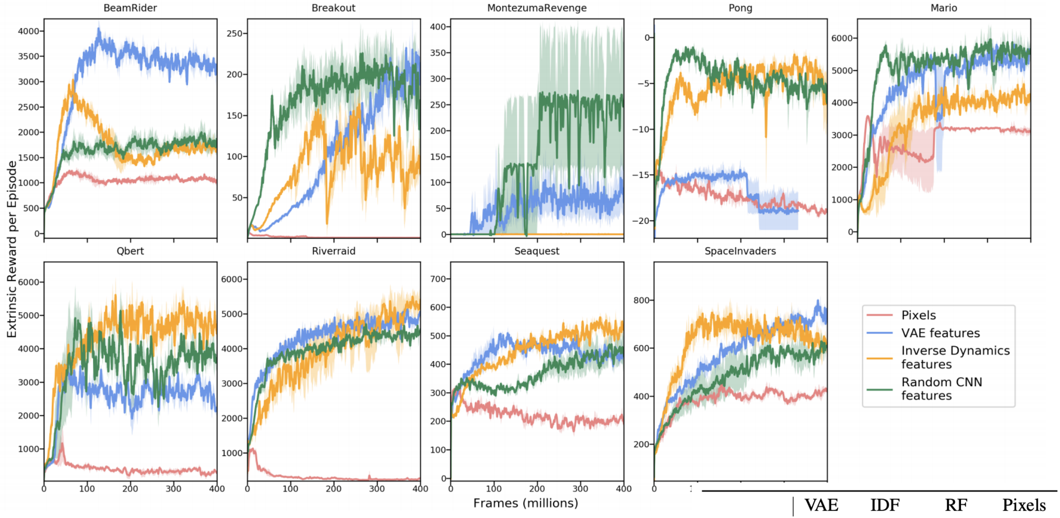
$$r_t = r_t^i = \|f(s_t, a_t) - \phi(s_{t+1})\|_2^2 \approx -\log p(s_{t+1}|s_t, a_t)$$

- 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

Results

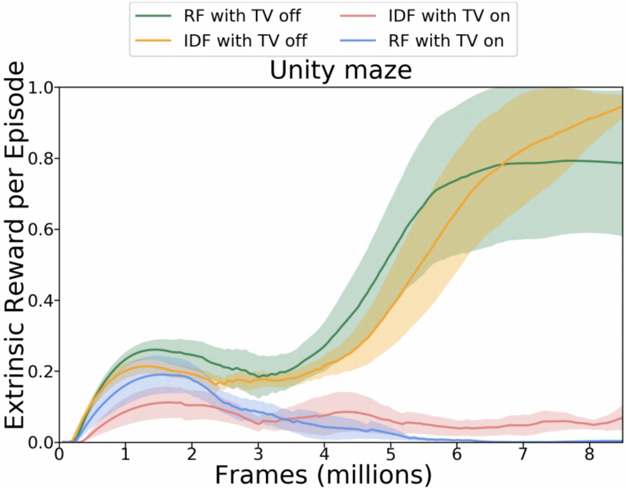


	VAE	IDF	RF	Pixels
Stable	No	No	Yes	Yes
Compact	Yes	Yes	Maybe	No
Sufficient	Yes	Maybe	Maybe	Yes

👍 *RF works quite well!*

(img from [Burda et al., 2019a])

Results: noisy TV



(img from [Burda et al., 2019a])

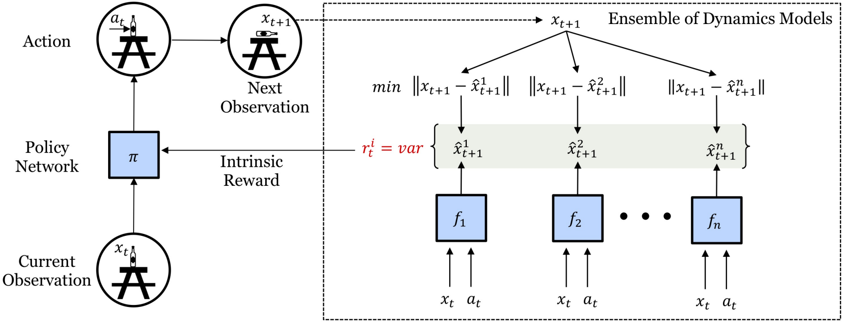
From One Model to Many

- All the methods used a single model to predict forward or inverse dynamics
- We can also use *multiple models* and *leverage disagreement*
  high disagreement $\implies$ low confidence $\implies$ need more data (*exploration*)

Self-Supervised Exploration via Disagreement

[Pathak et al., 2019]

- Ensemble method using multiple forward models (K models)



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

👍 *differentiable intrinsic reward*

👍 *can be paired with representation learning*

👎 *limitations of forward model learning*

(img from [Pathak et al., 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 $\implies$ *value predictions*

DORA
[Fox et al., 2018]

Consider two MDPs

- *Original MDP* $M = (S, A, p, r, \gamma)$
  $\implies$ learn $Q^\star_M(s,a)$ (*task objective*)

- *Cloned MDP* $M' = (S, A, p, 0, \gamma')$
  $\implies$ learn $E^\star(s,a) := Q^\star_{M'}(s,a) = 0$ (exploration)

Exploration bonus

$$r^i_t = \sqrt{\frac{1}{-\log E_t(s_t, a_t)}}$$

DORA

[Fox et al., 2018]

Consider two MDPs

- *Original MDP* $M = (S, A, p, r, \gamma)$
  $\implies$ learn $Q^\star_M(s,a)$ (*task objective*)
- *Cloned MDP* $M' = (S, A, p, 0, \gamma')$
  $\implies$ learn $E^\star(s,a) := Q^\star_{M'}(s,a) = 0$ (exploration)

Exploration bonus

$$r_t^i = \sqrt{\frac{1}{-\log E_t(s_t, a_t)}}$$

Idea:

- learn $E^\star$ online starting from $E_0(s,a) = 1$
- The E-value should converge to 0 ($E_k \to 0$)
- Then, $E_k(s,a) > 0$ *represents the prediction error*, i.e., uncertainty about the value of state (s,a)
- $\log E$ can be seen as a *generalized count*

use any preferred method to learn E with function approximation

Random Network Distillation (RND)

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

$$f_{\theta_*} : \mathcal{S} \to \mathbb{R}; \qquad f_\theta : \mathcal{S} \to \mathbb{R}$$

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

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

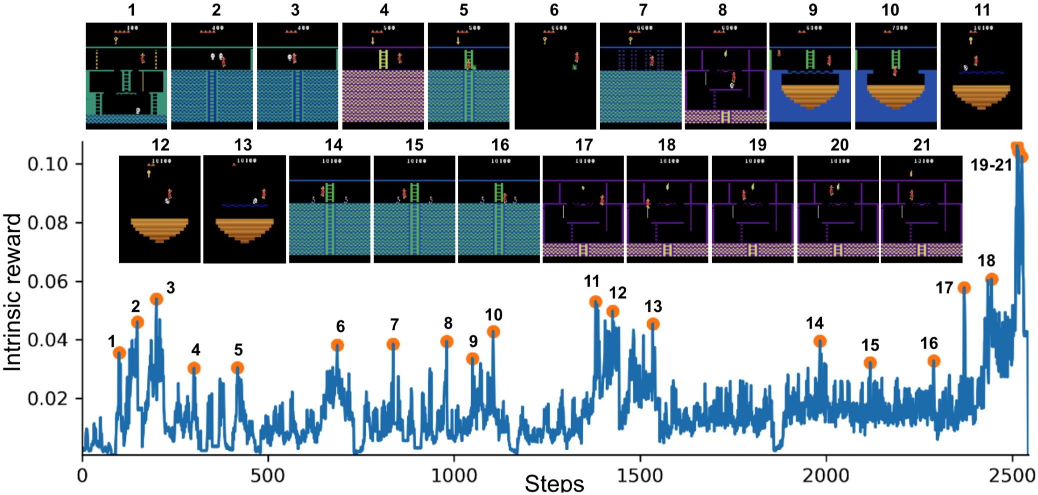
- Build "intrinsic" reward

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

👍 No model misspecification (f_θ can exactly predict f_{θ_*})

👍 Influence of learning dynamics can be reduced

Random Network Distillation (RND)



(img from [Burda et al., 2019b])

Random Network Distillation (RND)

[Burda et al., 2019b]

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. $V^I + cV^E$

"Tricks"

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

Random Network Distillation (RND)

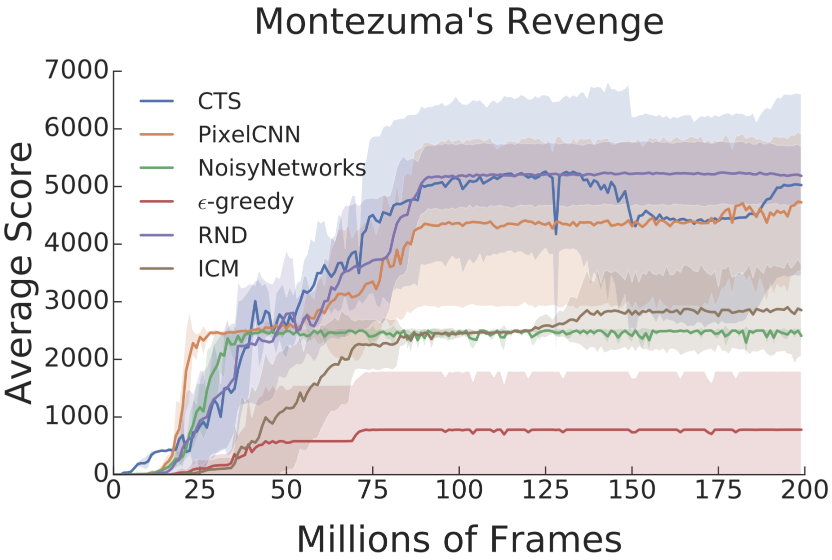
[Burda et al., 2019b]

Montezuma!

finds 22 out of the 24 rooms on the first level

Comparison

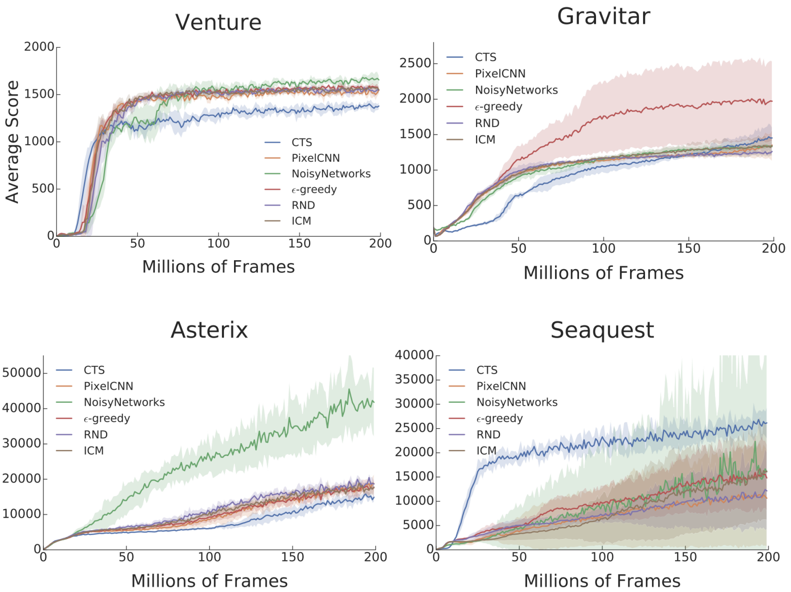
[Taïga et al., 2019]



(img from [Taïga et al., 2019])

Comparison: not all problems require same amount of exploration

[Taïga et al., 2019]



(img from [Taïga et al., 2019])

Exploration in Deep RL: Outline

Exploration Bonus: *Issues*

- Non-stationary
- Controllability/Predictability
  "agent finds a way to instantly gratify itself by exploiting actions which lead to hardly predictable consequences" – [Savinov et al., 2019]
- Knowledge fading
  "after the novelty of a state has vanished, the agent is not encouraged to visit it again, regardless of the downstream learning opportunities it might allow" – [Badia et al., 2020b]
- Representation learning intertwined with exploration

Few of these problems are addressed through *memory* (i.e., buffer)

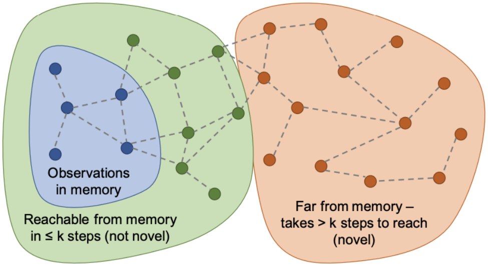
Exploration in Deep RL: Outline

Episodic Curiosity

[Savinov et al., 2019]

- The novelty bonus r_t^i depends on *reachability* of states
  i.e., give a reward only for those observations which take some effort to reach (*outside the known region*)

- Reachability = # steps between states

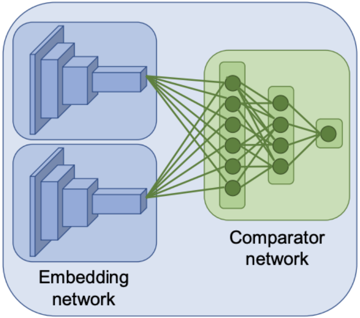


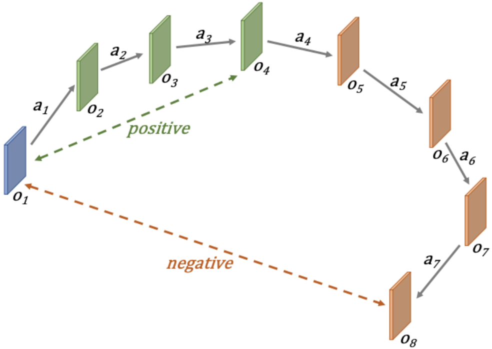
(img from [Savinov et al., 2019])

Components:

1. State embedding
2. Comparator
   (i.e., reachability predictor)
3. Episodic Memory

Episodic Curiosity: *Embedding* and *Reachability*





- Reachability is formulated as *binary classification problem*

$$C\Big(\phi(s_i), \phi(s_j)\Big) \mapsto [0, 1]$$

Episodic Curiosity: *Memory*

- Stores embeddings of past observations from the *current episode*

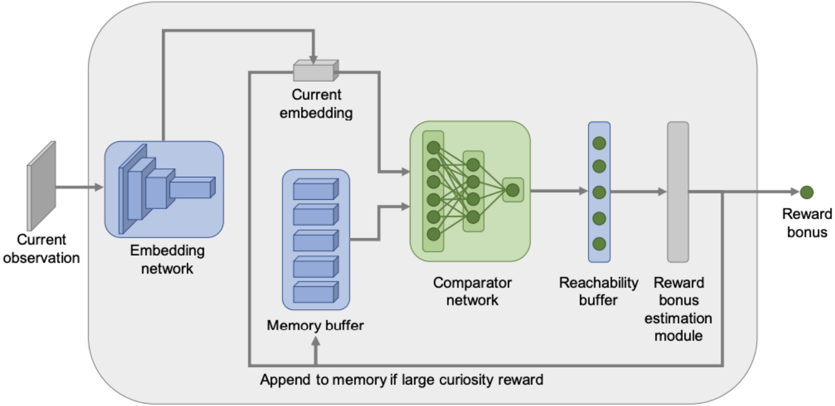
$$M = \left\{\phi(s_t)\right\}_t$$

- *Reinitialized* at the beginning of each episode
- Limited capacity
- $\phi(s_t)$ is added to M *only if novelty (bonus) is high enough*

Episodic Curiosity: *Bonus*

- $C(M, \phi(s_t))$ *similarity score* between the memory buffer and the current embedding (*may depend on all samples in* M)
- α and β are hyper-paramenters

$$r_t^i = \alpha(\beta - C(M, \phi(s_t)))$$



(img from [Savinov et al., 2019])

Vanishing Novelty? *Never Give Up*

$$r_t = \underbrace{r_t^e}_{\text{extrinsic reward (standard)}} + \beta \underbrace{r_t^i}_{\text{intrinsic}}$$

Intrinsic reward should capture [Badia et al., 2020b]

1. *Long-term novelty*
   reward encourages visiting states throughout training (*across episodes*)
2. *Short-term novelty*
   reward encourages visiting states over a short horizon (e.g., *within an episode*)
   ignores inter-episode interactions

Never Give Up: *intrinsic reward*

[Badia et al., 2020b]

$$r_t^i = \underbrace{r_t^{episodic}}_{\text{per-episode novelty (short-term)}} \cdot \min\left\{\max\{\underbrace{\alpha_t}_{\text{life-long novelty (long-term)}}, 1\}, L\right\}$$

Properties:

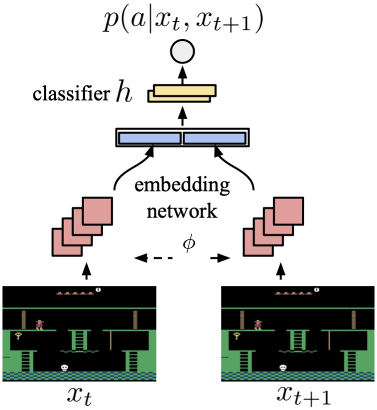
- *Rapidly discourages* revisiting states in an episode
- *Slowly discourages* revisiting frequent states across episodes

Never Give Up: *short-term* novelty

- **Episodic memory**: to store the *controllable* states in an online fashion

$$M = \left\{\phi(s_0), \phi(s_1), \ldots, \phi(s_{t-1})\right\}$$

- ϕ is an IDF (inverse dynamics features) *embedding* of the observation
  same as feature encoding in ICM



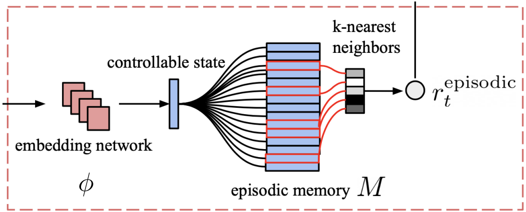
(img from [Badia et al., 2020b])

Never Give Up (NGU): *short-term* novelty

- *Frequency-based* exploration inside the **episode**

$$r_t^{\text{episodic}} = \frac{1}{\sqrt{n(\phi(s_t))}} \approx \frac{1}{\sqrt{\sum_{\phi_i \in N_k^t} Ker(\phi(s_t), \phi_i)} + c}$$

with N_k^t being the k-nearest neighbors of $\phi(s_t)$ in memory M



* $Ker(x,y) = \frac{\epsilon}{\frac{d^2(x,y)}{d_m^2} + \epsilon}$ where $d = \ell_2$ and d_m is a running average of the squared ℓ_2 distance of the k nearest neighbors

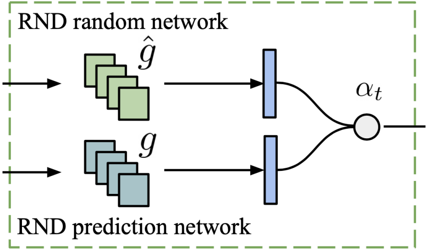
(img from [Badia et al., 2020b])

Never Give Up (NGU): *long-term* novelty

- *Random Network Distillation* [Burda et al., 2019b]

$$\alpha_t = 1 + \frac{err^{\mathrm{RND}}(s_t) - \mu_e}{\sigma_e}$$

σ_e and μ_e are running standard deviation and mean for $err^{\mathrm{RND}}(s_t)$



(img from [Badia et al., 2020b])

Never Give Up (NGU): *population training*

Multi-task setting: (population based, auxiliary tasks, ...)

- Learn *simultaneously* a family of problems (M_j) by approximating $Q(s, a; M_j)$
- (M_i) same dynamics but *different rewards*

$$r_t^{M_j} = r_t^e + \beta_j r_t^i$$

  with $\beta_0 = 0 < \ldots < \beta_{N-1} = \beta_{\max}$

- and *discount factors* $\gamma_0 = \gamma > \ldots > \gamma_{N-1}$
  in the paper $\gamma_0 = 0.997$, and $\gamma_{N-1} = 0.99$

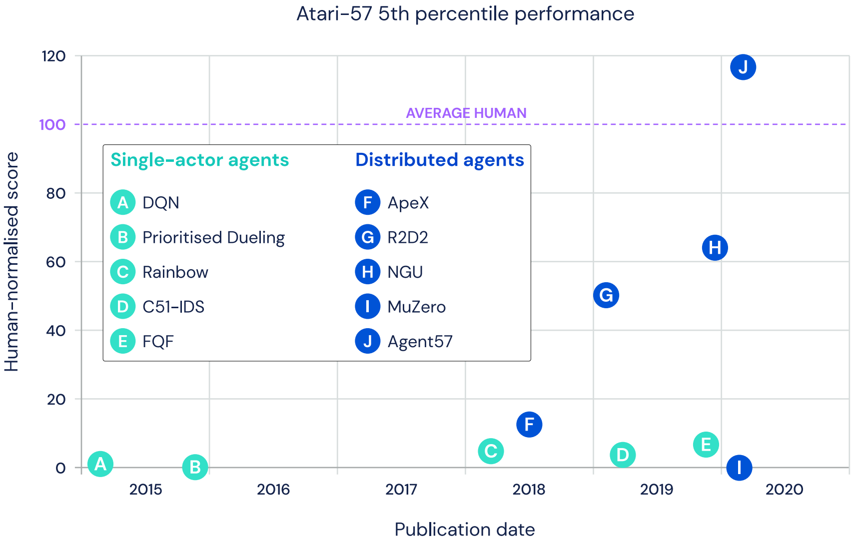
Beyond NGU: Agent57

[Badia et al., 2020a]

Agent57 *builds on NGU* but uses a new

1. State-Action Value Function Parameterization
2. Adaptive Exploration over a Family of Policies (*meta controller*)

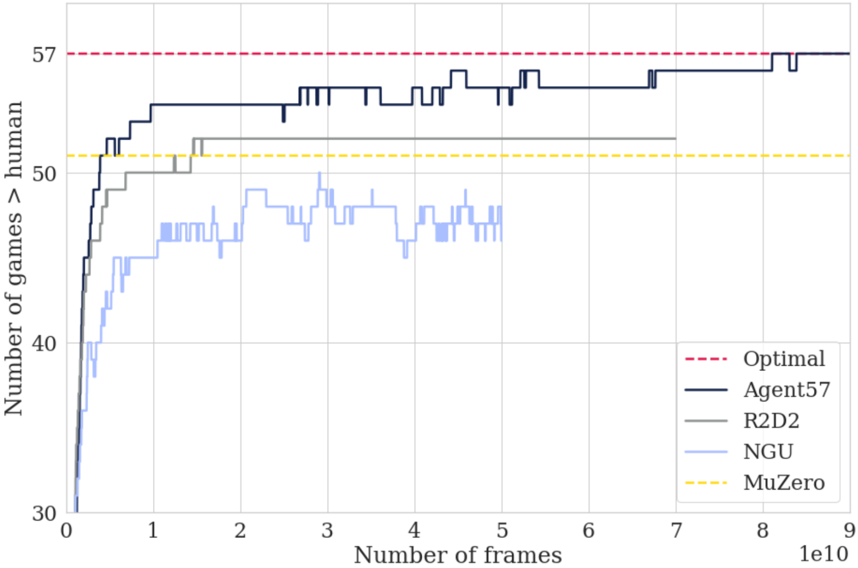
Super-human Performance



"[Agent57 is the] first deep reinforcement learning agent to obtain a score that is above the human baseline on all 57 Atari 2600 games."

(img from DM blog)

Frames to human performance



(img from [Badia et al., 2020a])

Agent57: *changes in Q*

Reparametrization

$$Q(s,a,j;\theta) = \underbrace{Q(s,a,j;\theta^e)}_{\text{extrinsic}} + \beta_j \underbrace{Q(s,a,j;\theta^i)}_{\text{intrinsic}}, \qquad \theta = \{\theta^e, \theta^i\}$$

- Q^e and Q^i have identical architecture
- Optimized using *transformed Retrace loss* (as NGU)
  - Optimized separately based on r^e and r^i respectively
  - But same target policy $\pi(s) = \arg\max_a Q(s,a,j;\theta)$

* new compared to NGU. Note that [Burda et al., 2019b] used two heads for extrinsic and intrinsic value function, with a shared architecture.

Agent57: *meta-controller*

NGU issue:

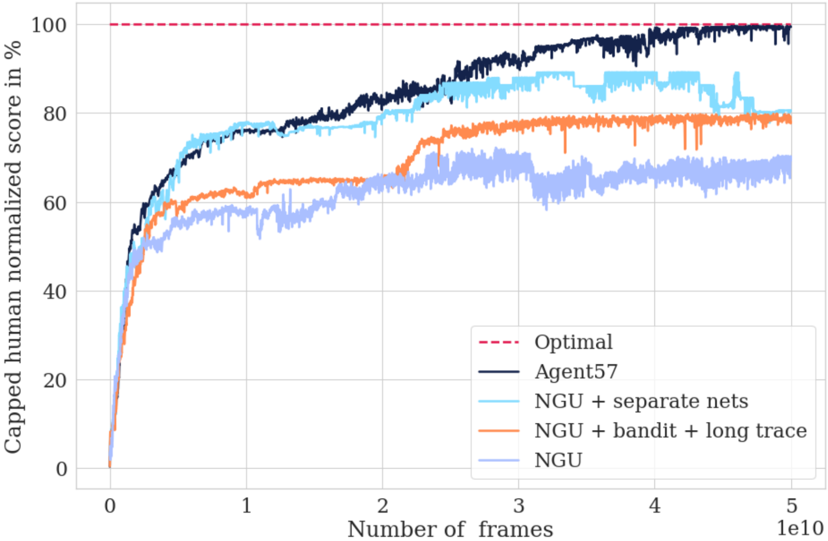
- all policies (i.e., models (M_j)) are trained equally, *regardless* of their contribution to the learning progress
- expected that higher β_j and lower γ_j do better in the early stages, and opposite later

Solution:

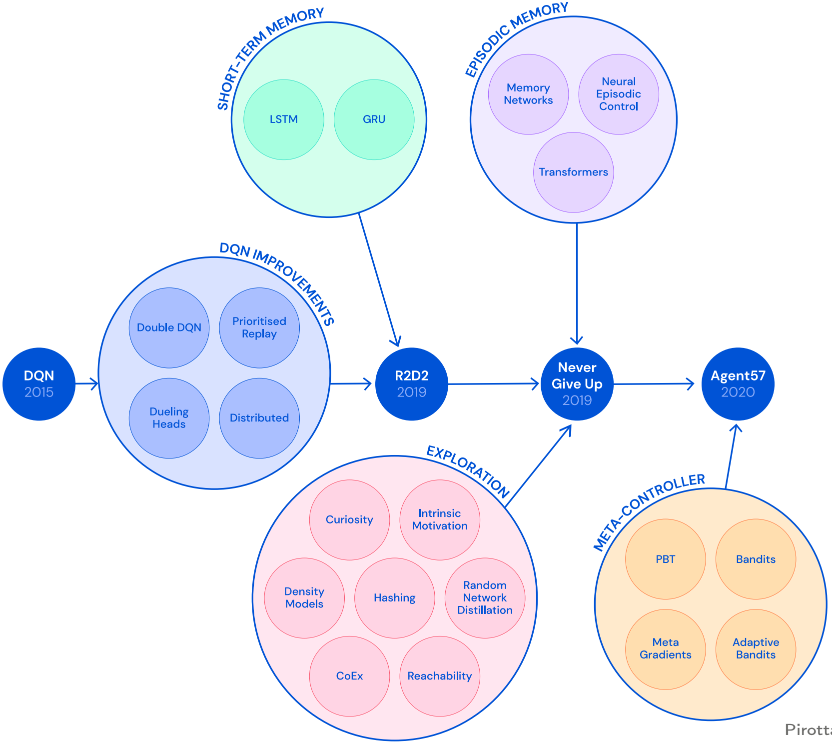
- meta-controller to prioritize what to learn
  $\implies$ sort of automatic curriculum learning
- use *non-stationary multi-arm bandit* algorithm [e.g., sliding-window UCB]
- *non-stationary?* Agent57 is also learning the policy of each task

Performance on 10 hard games

six hard *exploration* games, plus games that require *long-term credit assignment*.
Beam Rider, Freeway, Montezuma's Revenge, Pitfall!, Pong, Private Eye, Skiing, Solaris, Surround, and Venture



Video



(img adapted from DM blog)

Pirotta

Exploration in Deep RL: Outline

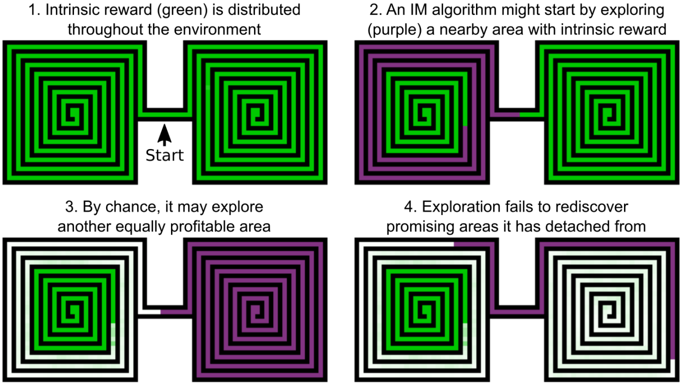
Direct Exploration

[Ecoffet et al., 2019, 2021]

Recall a few issues of intrinsic exploration:

- Forget about promising areas they have visited
- They do not return to them for further exploration

Green areas indicate intrinsic reward, white indicates areas where no intrinsic reward remains, and purple areas indicate where the algorithm is currently exploring.



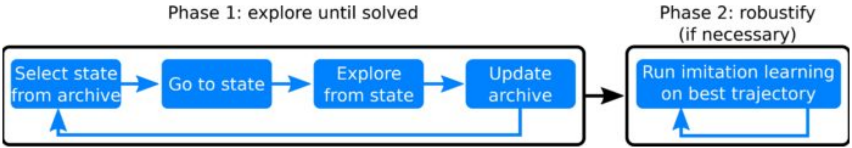
(img from [Ecoffet et al., 2019])

It would be good to keep in memory unexplored states and target them

Go-Explore: *Phases*

[Ecoffet et al., 2019, 2021]

1. Exploration
2. Robustification

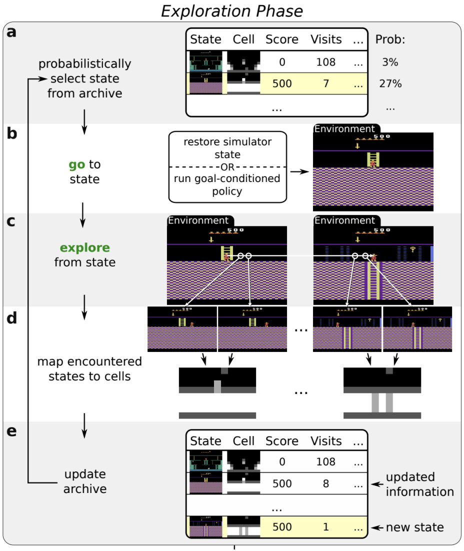


(img from [Ecoffet et al., 2019])

Go-Explore: *phases* ①

Steps:

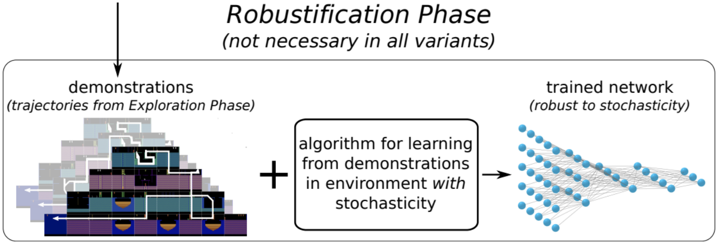
- Select a state $\phi(s)$ from memory M
  e.g., by relevance (IM, novelty, etc.)
- Go to a state $\phi(s)$
- Explore locally (e.g., randomly)
- Store embedding $\phi(s')$ in M



(img from https://arxiv.org/pdf/2004.12919.pdf)

Go-Explore: *phases* ②

- Robustification against noise
- Learning from Demonstrations (i.e., imitation learning)
  requires to store the highest-scoring trajectories



(img from https://arxiv.org/pdf/2004.12919.pdf)

Intuition: discover and control

- **Discover states**
  This is done by random exploration around the targeted state

- **Control states**
  This is obtained by the incremental approach

⚠ *Most challenging aspect is reaching the selected state in phase ①*

* similar to theoretical approaches for autonomous exploration [e.g., Lim and Auer, 2012, Tarbouriech et al., 2020]

Reaching a State

[Ecoffet et al., 2019, 2021]

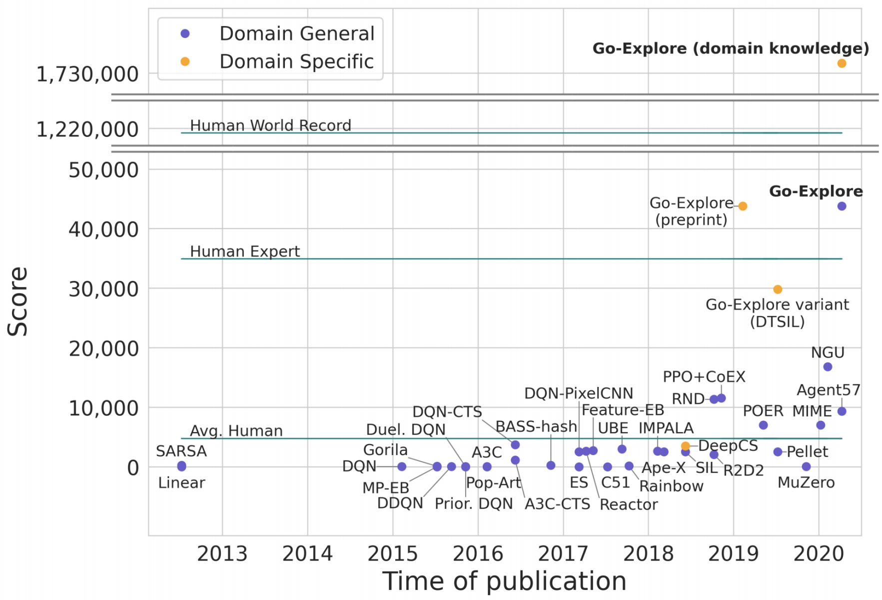
Ⓐ – *Resetting the simulator*

- Strong assumption
- Leverage determinism through the simulator
- Based on replaying actions

Ⓑ – *Goal-Oriented policy*

- Generic setting
- Learn policies aiming to reach a specific state
  learn goal-dependent quantities, e.g., $Q(s, a; g)$ or $\pi(s, a; g)$
- They train $\pi(s, a; g)$ based on the best trajectory that led to such a goal g + imitation-learning

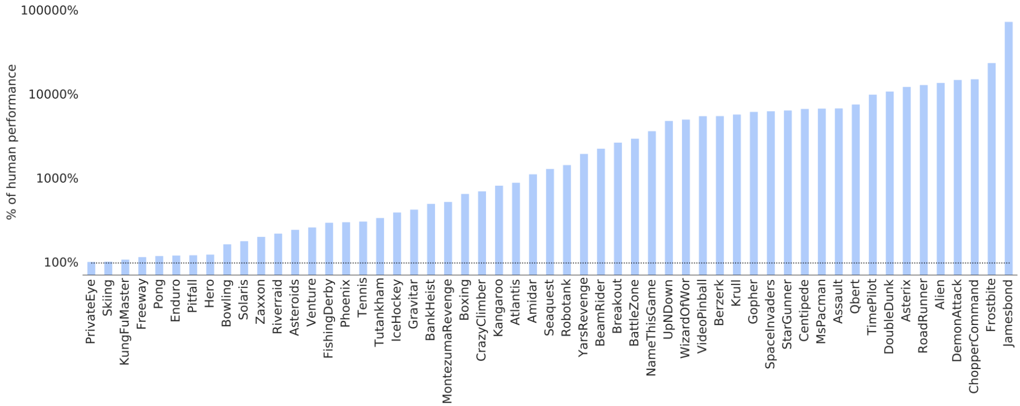
Results with Simulator Reset



(a) Historical progress on Montezuma's Revenge.

(img from https://arxiv.org/pdf/2004.12919.pdf)

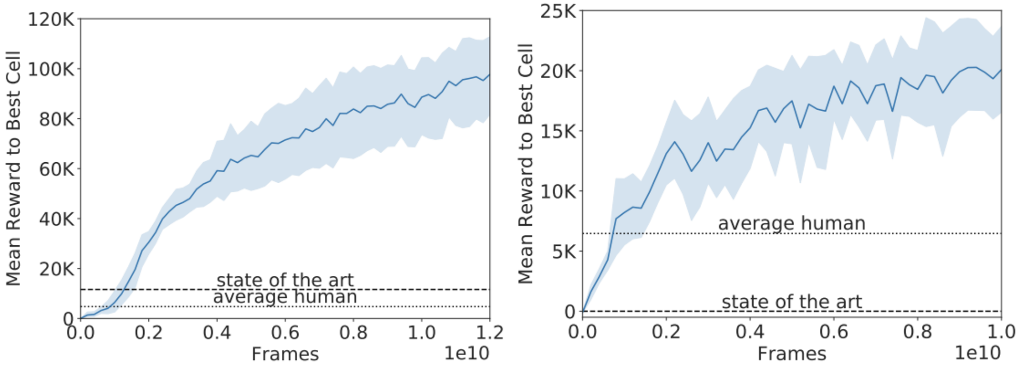
Results with Simulator Reset - cont'd



(img from https://arxiv.org/pdf/2004.12919.pdf)

Video

Results with Goal-Oriented Policy



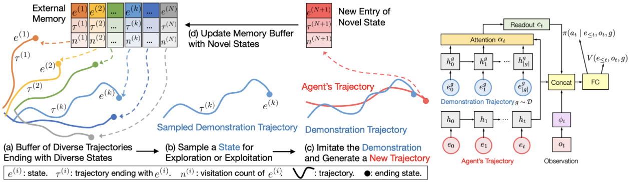
(a) Montezuma's Revenge

(b) Pitfall

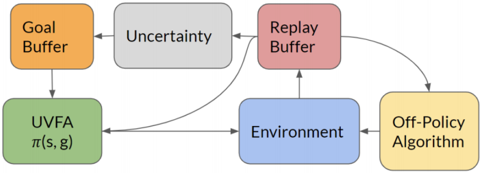
(img from https://arxiv.org/pdf/2004.12919.pdf)

Other Approaches for Direct Exploration

- [Guo et al., 2020] (DTSIL)
  keep trajectories and train a goal/trajectory oriented policy by imitation learning



- [Guo and Brunskill, 2019]
  learn goal-conditioned policy to directly reach highly-uncertain states



Exploration in Deep RL: Outline

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 $\widetilde{\theta} = \theta + \text{noise}$
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
$\implies$ long-term exploration
$\implies$ no dithering

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 $\widetilde{\theta} = \theta + \text{noise}$
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
$\implies$ long-term exploration
$\implies$ no dithering

The randomness needs to represent "uncertainty"

Exploration via Randomization

- Perturb observed rewards
  store samples $(s, a, s', r + \mathrm{noise})$ 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., 2019, 2018, Azizzadenesheli et al., 2018, Lipton et al., 2018, Touati et al., 2019, Osband et al., 2019]

RVF: *issues*

Reward Perturbation

- Minimize least-squares problem for any reward structure
  e.g., by gradient descent
- Not so easy to define the magnitude of the reward perturbation

Posterior Sampling

- Posterior variance
  - easy for linear model
  - hard (almost impossible) for generic models
- A lot of approximate schemas for computing the posterior

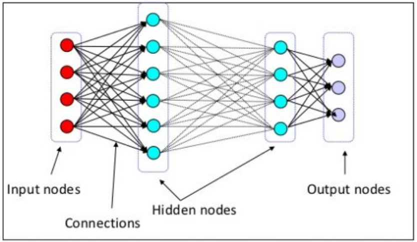
Posterior Distribution for Deep Neural Networks

Bayesian DQN [Azizzadenesheli et al., 2018]

1. Bayesian linear regression with given feature $\phi(s) \in \mathbb{R}^d$ and given target vector for each action y_a

$$\mu_a = (\Phi_a^\mathsf{T}\Phi_a)^{-1}\Phi_a^\mathsf{T} y_a \qquad \Sigma_a = \Phi_a^\mathsf{T}\Phi_a$$

2. Draw a weight vector at random $w_a \sim \mathcal{N}(\mu_a, \Sigma_a^{-1})$
3. Run the corresponding (greedy) policy
$a_t = \arg\max_a Q(s_t, a) := \arg\max_a w_a^\mathsf{T}\phi(s_t)$
4. Train ϕ with standard NN to estimate Q



⚠ Same tools as in linear bandit

Posterior Distribution for Deep Neural Networks

BBQ-Networks [Lipton et al., 2018]

- Uses variational inference to quantify uncertainty
- Uses independent factorized Gaussians as an approximate posterior

MNF-DQN [Touati et al., 2019]

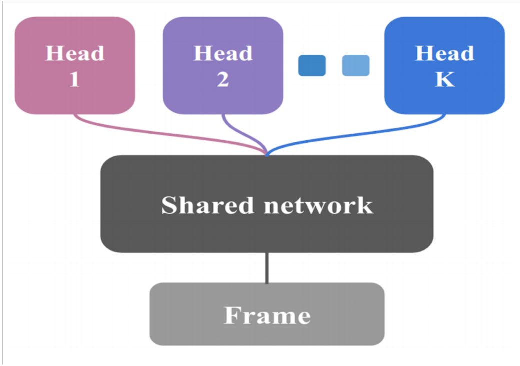
- Leverages recent advances in variational Bayesian NN
- Computationally and statistically efficient
- Uses normalizing multiplicative flows (MNF) in order to account for the uncertainty of estimates for efficient exploration

Bootstrap DQN

[Osband et al., 2016]

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

Bootstrap DQN

[Osband et al., 2016]

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 ~ Uniform{1, ..., K}
    for step t = 1, ... until end of episode do
        Pick an action according to a_t ∈ argmax_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 ~ M
        Add (s_t, a_t, r_{t+1}, s_{t+1}, m_t) to replay buffer B
    end for
end for
```

- 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 $y_t = r_t + \max_a Q(s_{t+1}, a; \theta^-)$

Randomized Prior Functions

[Osband et al., 2018]

Bayesian perspective: "generate posterior samples by training on noisy versions of the data, together with some random regularization"

Randomized Prior + Bootstrapped DQN

- Train an ensemble of models, each on *perturbed versions of the data*
- The resulting distribution of the ensemble is used to approximate the uncertainty in the estimate

$$\mathcal{L}(\theta; \theta^{-}, p, \mathcal{D}) = \sum_{t \in \mathcal{D}} \left(r_t + \gamma \max_{a'} (Q_{\theta^-} + p)(s'_t, a') - (Q_\theta + p)(s_t, a_t) \right)$$

Noisy Networks

[Fortunato et al., 2018]

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

$$w \to \mu^w, \sigma^w \text{ and } b \to \mu^b, \sigma^b$$

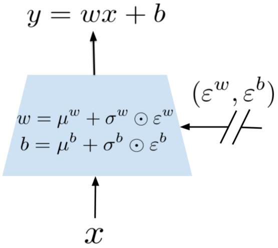
- Whenever a layer is evaluated draw $\varepsilon^w, \varepsilon^b \sim \mathcal{D}$
- Evaluate the "random" layer as

$$y = (\mu^w + \sigma^w \odot \varepsilon^w) + \mu^b + \sigma^b \odot \varepsilon^b$$

- Let $\zeta = (\mu^w, \sigma^w, \mu^b, \sigma^b)$, define the expected loss

$$\overline{L}(\zeta) = \mathbb{E}_\varepsilon\left[L(\zeta, \varepsilon)\right]$$

- Gradient estimation update



Noisy Networks: *noise models*

[Fortunato et al., 2018]

- Independent noise $\varepsilon_{i,j}$ for each weight i at layer j
- Factorized noise $\varepsilon_{i,j} = f(\varepsilon_i) f(\varepsilon_j)$ (e.g., $f(x) = \mathrm{sgn}(x)\sqrt{x}$)
- Independent noise for target and online networks

$$y_t = r_t + \max_{a'} Q(s'_t, a'; \varepsilon', \zeta^-); \qquad L_t(\zeta, \varepsilon) = \big(y_t - Q(s_t, a_t; \varepsilon, \zeta)\big)^2$$

Comparison

[Touati et al., 2019]

Simple Chain domain

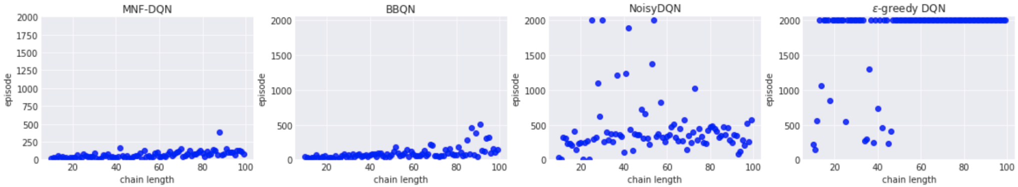
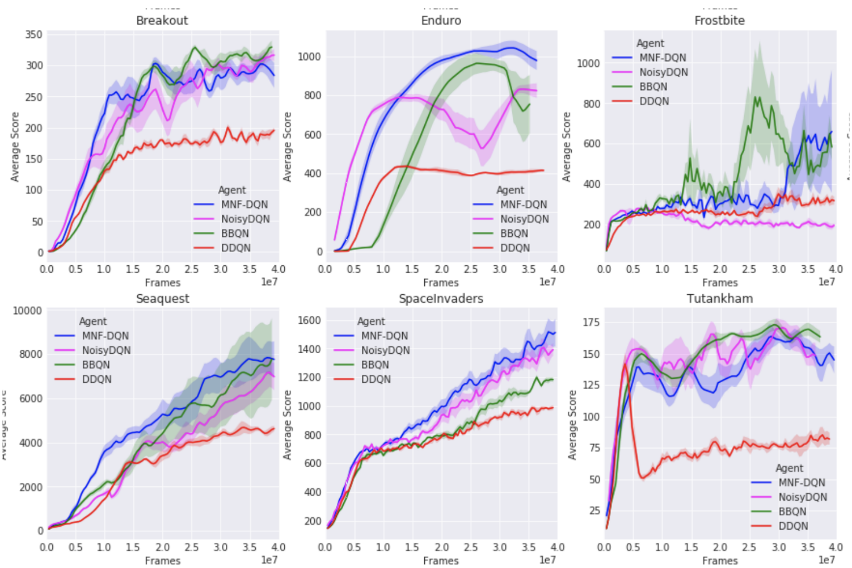


Figure 1: Median number of episodes (max 2000) required to solve the n-chain problem for (figure from left to right) MNF-DQN, BBQN, NoisyDQN and ϵ-greedy DQN. The median is obtained over 10 runs with different seeds. We see that MNF-DQN consistently performs best across different chain lengths.

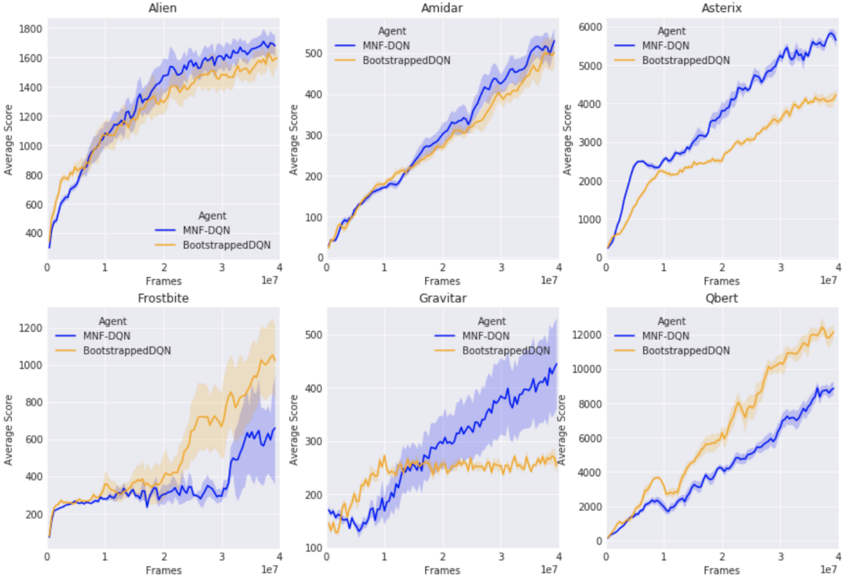
Comparison: Atari

[Touati et al., 2019]



Comparison: Atari

[Touati et al., 2019]



Exploration in Deep RL: Outline

Exploration in Deep RL

- Several different techniques (*we have see 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?

- Information Gain
- Exploration via options
- Task-agnostic exploration
- Multi-task settings
- Meta learning

Thank you

Materials

- References in these slides
- This nice blog post about exploration in DeepRL

POLITEX: regret bounds for policy iteration using expert prediction. In Yasin Abbasi-Yadkori, Peter L. Bartlett, Kush Bhatia, Nevena Lazic, Csaba Szepesvári, and Gellért Weisz, editors, *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 3692–3702. PMLR, 2019.

Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 263–272. PMLR, 2017.

Kamyar Azizzadenesheli, Emma Brunskill, and Animashree Anandkumar. Efficient exploration through bayesian deep q-networks. In *ITA*, pages 1–9. IEEE, 2018.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 507–517. PMLR, 2020a.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andrew Bolt, and Charles Blundell. Never give up: Learning directed exploration strategies. In *ICLR*. OpenReview.net, 2020b.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying count-based exploration and intrinsic motivation. In *NIPS*, pages 1471–1479, 2016.

Yuri Burda, Harrison Edwards, Deepak Pathak, Amos J. Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In *ICLR (Poster)*. OpenReview.net, 2019a.

Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. In *ICLR (Poster)*. OpenReview.net, 2019b.

Qi Cai, Zhuoran Yang, Chi Jin, and Zhaoran Wang. Provably efficient exploration in policy optimization. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 1283–1294. PMLR, 2020.

Christoph Dann, Tor Lattimore, and Emma Brunskill. Unifying PAC and regret: Uniform PAC bounds for episodic reinforcement learning. In *NIPS*, pages 5713–5723, 2017.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *CoRR*, abs/1901.10995, 2019.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

Yonathan Efroni, Nadav Merlis, Mohammad Ghavamzadeh, and Shie Mannor. Tight regret bounds for model-based reinforcement learning with greedy policies. In *NeurIPS*, pages 12203–12213, 2019.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Rémi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. In *ICLR (Poster)*. OpenReview.net, 2018.

Lior Fox, Leshem Choshen, and Yonatan Loewenstein. DORA the explorer: Directed outreaching reinforcement action-selection. In *ICLR (Poster)*. OpenReview.net, 2018.

Ronan Fruit, Matteo Pirotta, and Alessandro Lazaric. Near optimal exploration-exploitation in non-communicating markov decision processes. In *NeurIPS*, pages 2998–3008, 2018a.

Ronan Fruit, Matteo Pirotta, Alessandro Lazaric, and Ronald Ortner. Efficient bias-span-constrained exploration-exploitation in reinforcement learning. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 1573–1581. PMLR, 2018b.

Ronan Fruit, Matteo Pirotta, and Alessandro Lazaric. Improved analysis of UCRL2 with empirical bernstein inequality. *CoRR*, abs/2007.05456, 2020.

Hao Gong and Mengdi Wang. A duality approach for regret minimization in average-award ergodic markov decision processes. In *L4DC*, volume 120 of *Proceedings of Machine Learning Research*, pages 862–883. PMLR, 2020.

Yijie Guo, Jongwook Choi, Marcin Moczulski, Shengyu Feng, Samy Bengio, Mohammad Norouzi, and Honglak Lee. Memory based trajectory-conditioned policies for learning from sparse rewards. In *NeurIPS*, 2020.

Zhaohan Daniel Guo and Emma Brunskill. Directed exploration for reinforcement learning. *CoRR*, abs/1906.07805, 2019.

Botao Hao, Nevena Lazic, Yasin Abbasi-Yadkori, Pooria Joulani, and Csaba Szepesvári. Adaptive approximate policy iteration. In *AISTATS*, 2021.

Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *J. Mach. Learn. Res.*, 11:1563–1600, 2010.

Chi Jin, Zeyuan Allen-Zhu, Sébastien Bubeck, and Michael I. Jordan. Is q-learning provably efficient? In *NeurIPS*, pages 4868–4878, 2018.

Shiau Hong Lim and Peter Auer. Autonomous exploration for navigating in mdps. In *COLT*, volume 23 of *JMLR Proceedings*, pages 40.1–40.24. JMLR.org, 2012.

Zachary C. Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. In *AAAI*, pages 5237–5244. AAAI Press, 2018.

Pierre Menard, Omar Darwiche Domingues, Xuedong Shang, and Michal Valko. Ucb momentum q-learning: Correcting the bias without forgetting, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Gergely Neu and Ciara Pike-Burke. A unifying view of optimism in episodic reinforcement learning. In *NeurIPS*, 2020.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In *NIPS*, pages 4026–4034, 2016.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *NeurIPS*, pages 8626–8638, 2018.

Ian Osband, Benjamin Van Roy, Daniel J. Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019. URL http://jmlr.org/papers/v20/18-339.html.

Georg Ostrovski, Marc G. Bellemare, Aäron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 2721–2730. PMLR, 2017.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 2778–2787. PMLR, 2017.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 5062–5071. PMLR, 2019.

Jian Qian, Ronan Fruit, Matteo Pirotta, and Alessandro Lazaric. Exploration bonus for regret minimization in discrete and continuous average reward mdps. In *NeurIPS*, pages 4891–4900, 2019.

Nikolay Savinov, Anton Raichuk, Damien Vincent, Raphaël Marinier, Marc Pollefeys, Timothy P. Lillicrap, and Sylvain Gelly. Episodic curiosity through reachability. In *ICLR (Poster)*. OpenReview.net, 2019.

Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990-2010). *IEEE Trans. Auton. Ment. Dev.*, 2(3):230–247, 2010.

Lior Shani, Yonathan Efroni, Aviv Rosenberg, and Shie Mannor. Optimistic policy optimization with bandit feedback. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 8604–8613. PMLR, 2020.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Bradly C. Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *CoRR*, abs/1507.00814, 2015.

Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

Yifan Sun, Yaqi Duan, Hao Gong, and Mengdi Wang. Learning low-dimensional state embeddings and metastable clusters from time series data. In *NeurIPS*, pages 4563–4572, 2019.

Adrien Ali Taïga, William Fedus, Marlos C. Machado, Aaron C. Courville, and Marc G. Bellemare. Benchmarking bonus-based exploration methods on the arcade learning environment. *CoRR*, abs/1908.02388, 2019.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. #exploration: A study of count-based exploration for deep reinforcement learning. In *NIPS*, pages 2753–2762, 2017.

Jean Tarbouriech, Matteo Pirotta, Michal Valko, and Alessandro Lazaric. Improved sample complexity for incremental autonomous exploration in mdps. In *NeurIPS*, 2020.

Ahmed Touati, Harsh Satija, Joshua Romoff, Joelle Pineau, and Pascal Vincent. Randomized value functions via multiplicative normalizing flows. In *UAI*, page 156. AUAI Press, 2019.

Chen-Yu Wei, Mehdi Jafarnia-Jahromi, Haipeng Luo, Hiteshi Sharma, and Rahul Jain. Model-free reinforcement learning in infinite-horizon average-reward markov decision processes. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 10170–10180. PMLR, 2020.

Andrea Zanette and Emma Brunskill. Problem dependent reinforcement learning bounds which can identify bandit structure in mdps. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 5732–5740. PMLR, 2018.

Andrea Zanette and Emma Brunskill. Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 7304–7312. PMLR, 2019.

Zihan Zhang and Xiangyang Ji. Regret minimization for reinforcement learning by evaluating the optimal bias function. In *NeurIPS*, pages 2823–2832, 2019.

Zihan Zhang, Yuan Zhou, and Xiangyang Ji. Almost optimal model-free reinforcement learning via reference-advantage decomposition. In *NeurIPS*, 2020.