Exploration Strategies

Bibek Poudel

ECE 414/ 517 Reinforcement Learning

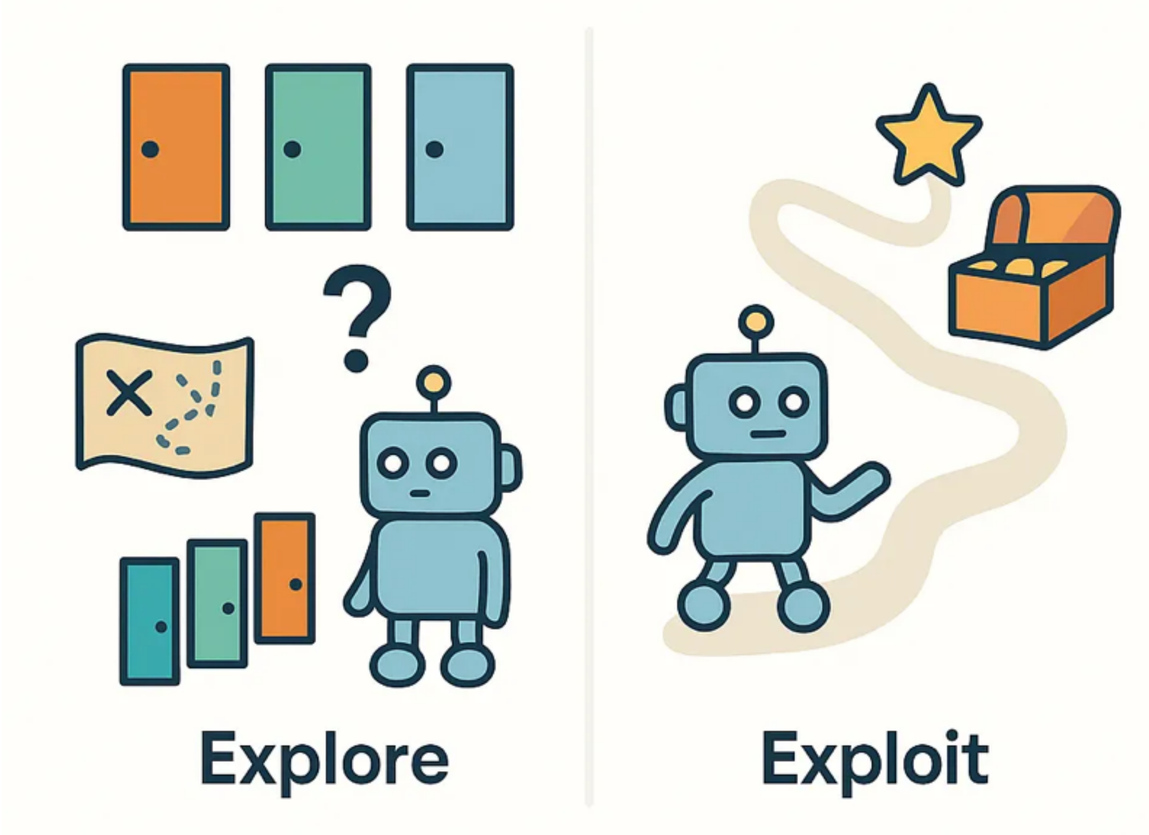
Outline

- Exploration vs Exploitation Dilemma
- ε-Greedy Action Selection
- Boltzmann Exploration
- Upper Confidence Bound
- Entropy Regularization
- Random Network Distillation

Exploration vs Exploitation Dilemma

Exploration vs Exploitation Dilemma

- **Explore:** Gather more information
- **Exploit:** Make the best decision given the current information

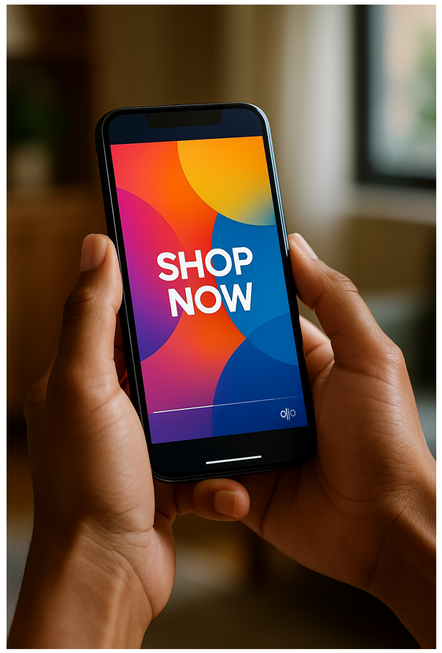


https://pub.towardsai.net/what-is-reinforcement-learning-a-deep-and-practical-guide-to-the-most-powerful-idea-in-ai-m006-56144a292de9

Exploration vs Exploitation Dilemma

Restaurant

Advertisement



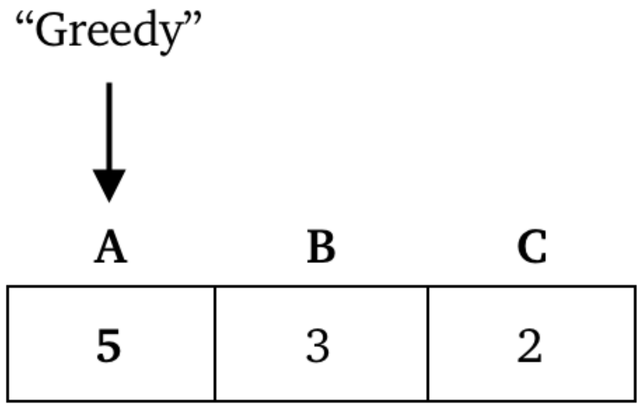
Exploration vs Exploitation Dilemma

- Agent maintains action values

A	B	C
5	3	2

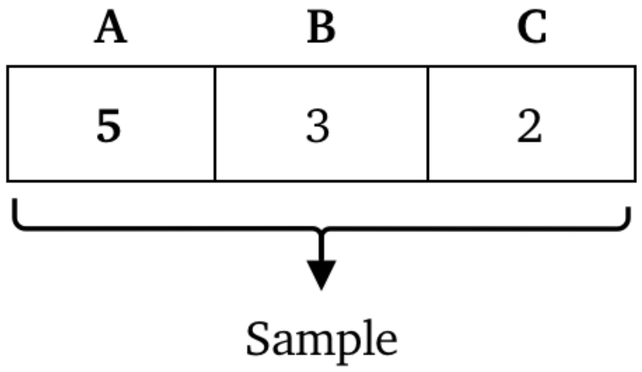
Exploration vs Exploitation Dilemma

- Agent maintains action values
- Exploitation = Select the greedy action



Exploration vs Exploitation Dilemma

- Agent maintains action values
- Exploration = Sample from the action space



Exploration vs Exploitation Dilemma

- So why the “dilemma”/ “trade-off”/ “conflict”?
  - ✦ Because at a single time-step, the agent cannot do both.
- Why explore?

Exploration vs Exploitation Dilemma

- So why the “dilemma”/ “trade-off”/ “conflict”?
  - ✦ Because at a single time-step, the agent cannot do both.
- Why explore?
  - ✦ Online environment
  - ✦ Navigate uncertainty
  - ✦ Best long term strategy may involve short term sacrifices

ε-Greedy Action Selection

ε-Greedy Action Selection

- Random Exploration with a small probability ε

ε-Greedy Action Selection

Algorithm 1 ε-Greedy Action Selection

Require: state s, action set $\mathcal{A}$, estimates $Q(s, a)$, parameter $\varepsilon \in [0, 1]$

ε-Greedy Action Selection

Algorithm 1 ε-Greedy Action Selection

Require: state s, action set $\mathcal{A}$, estimates $Q(s, a)$, parameter $\varepsilon \in [0, 1]$

sample $u \sim \text{Uniform}(0, 1)$
if $u \leq \varepsilon$ **then** ▷ explore with probability ε
    $a \leftarrow \text{Random}(\mathcal{A})$

ε-Greedy Action Selection

Algorithm 1 ε-Greedy Action Selection

Require: state s, action set $\mathcal{A}$, estimates $Q(s, a)$, parameter $\varepsilon \in [0, 1]$

sample $u \sim \text{Uniform}(0, 1)$
if $u \leq \varepsilon$ **then** ▷ explore with probability ε
    $a \leftarrow \text{Random}(\mathcal{A})$
else
    $a \leftarrow \arg\max_{a' \in \mathcal{A}} Q(s, a')$
end if
return a

ε-Greedy Action Selection

Exercise: In ε-greedy action selection, for the case of two actions and $\varepsilon = 0.5$, what is the probability that the greedy action is selected?

Sutton & Barto, Reinforcement Learning, Second edition.

ε-Greedy Action Selection

Exercise: In ε-greedy action selection, for the case of two actions and $\varepsilon = 0.5$, what is the probability that the greedy action is selected?

Solution:

$$\varepsilon = 0.5, \text{ two actions } \{A, B\}, \text{ assume } A \text{ is greedy}$$

- $P(A) = (1 - \varepsilon) + \varepsilon \cdot P(A \mid \text{explore})$
- $P(A) = (1 - 0.5) + 0.5 \times \frac{1}{2} = 0.75$

ε-Greedy Action Selection

Exercise: In ε-greedy action selection, for the case of two actions and $\varepsilon = 0.5$, what is the probability that the greedy action is selected?

Solution:

$\varepsilon = 0.5$, two actions $\{A, B\}$, assume A is greedy

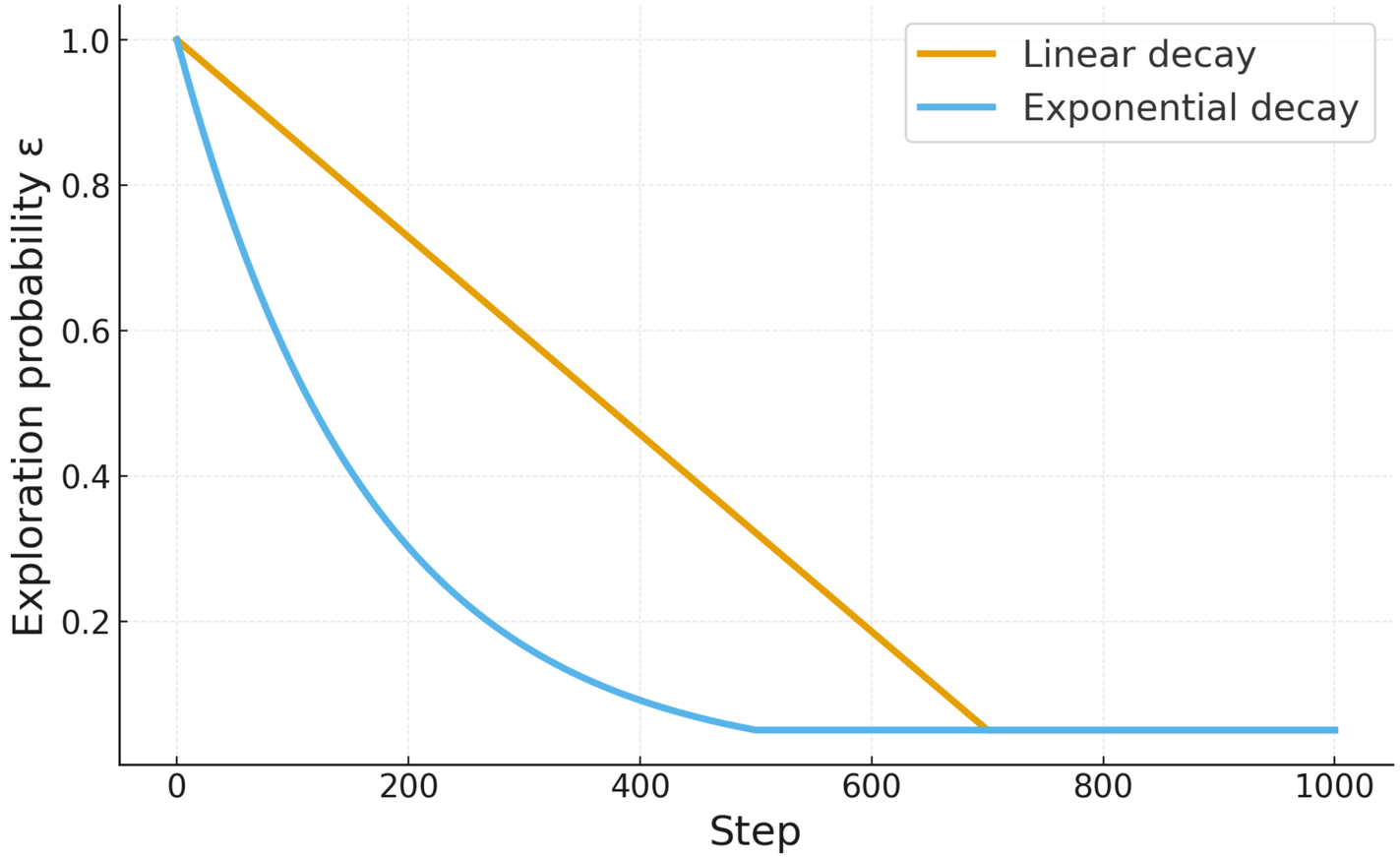
- $P(A) = (1 - \varepsilon) + \varepsilon \cdot P(A \mid \text{explore})$
- $P(A) = (1 - 0.5) + 0.5 \times \frac{1}{2} = 0.75$

`A` could be sampled during exploitation

`A` could also be sampled during exploration

ε-Greedy Action Selection

- Decay the ε
- Linear, exponential



Boltzmann Exploration

Boltzmann Exploration

- Apply Soft-max function over the Q values

Boltzmann Exploration

- Apply Soft-max function over the Q values

$$\pi(a|s) = \frac{e^{Q(s,a)}}{\sum_{b \in \mathcal{A}} e^{Q(s,b)}}$$

Boltzmann Exploration

- Apply Soft-max function over the Q values

Consider 4 actions with the following Q-values:

$$Q(s, a_1) = 2.0 \qquad Q(s, a_3) = 0.5$$
$$Q(s, a_2) = 1.0 \qquad Q(s, a_4) = -1.0$$

$$\pi(a|s) = \frac{e^{Q(s,a)}}{\sum_{b \in \mathcal{A}} e^{Q(s,b)}}$$

Boltzmann Exploration

- Apply Soft-max function over the Q values

Consider 4 actions with the following Q-values:

$$Q(s, a_1) = 2.0 \qquad Q(s, a_3) = 0.5$$
$$Q(s, a_2) = 1.0 \qquad Q(s, a_4) = -1.0$$

$$\pi(a|s) = \frac{e^{Q(s,a)}}{\sum_{b \in \mathcal{A}} e^{Q(s,b)}}$$

Standard Softmax

Step 1: Compute exponentials

$$e^{2.0} = 7.39 \qquad e^{0.5} = 1.65$$
$$e^{1.0} = 2.72 \qquad e^{-1.0} = 0.37$$

Step 2: Sum of exponentials

$$\sum_{b \in \mathcal{A}} e^{Q(s,b)} = 7.39 + 2.72 + 1.65 + 0.37 = 12.13$$

Boltzmann Exploration

- Apply Soft-max function over the Q values

Consider 4 actions with the following Q-values:

$$Q(s, a_1) = 2.0 \qquad Q(s, a_3) = 0.5$$
$$Q(s, a_2) = 1.0 \qquad Q(s, a_4) = -1.0$$

$$\pi(a|s) = \frac{e^{Q(s,a)}}{\sum_{b \in \mathcal{A}} e^{Q(s,b)}}$$

Standard Softmax

Step 1: Compute exponentials

$$e^{2.0} = 7.39 \qquad e^{0.5} = 1.65$$
$$e^{1.0} = 2.72 \qquad e^{-1.0} = 0.37$$

Step 2: Sum of exponentials

$$\sum_{b \in \mathcal{A}} e^{Q(s,b)} = 7.39 + 2.72 + 1.65 + 0.37 = 12.13$$

Step 3: Compute probabilities

$$\pi(a_1|s) = \frac{7.39}{12.13} = 0.61 \ (61\%) \qquad \pi(a_3|s) = \frac{1.65}{12.13} = 0.14 \ (14\%)$$
$$\pi(a_2|s) = \frac{2.72}{12.13} = 0.22 \ (22\%) \qquad \pi(a_4|s) = \frac{0.37}{12.13} = 0.03 \ (3\%)$$

Probabilities are proportional to Q !!

Boltzmann Exploration

- Apply Soft-max function over the Q values (with temperature)

$$\pi(a|s) = \frac{e^{Q(s,a)/\tau}}{\sum_{b \in \mathcal{A}} e^{Q(s,b)/\tau}}$$

Boltzmann Exploration

- Apply Soft-max function over the Q values (with temperature)

Consider 4 actions with the following Q-values:

$$Q(s, a_1) = 2.0 \qquad Q(s, a_3) = 0.5$$
$$Q(s, a_2) = 1.0 \qquad Q(s, a_4) = -1.0$$

$$\pi(a|s) = \frac{e^{Q(s,a)/\tau}}{\sum_{b \in \mathcal{A}} e^{Q(s,b)/\tau}}$$

Boltzmann Exploration

- Apply Soft-max function over the Q values (with temperature)

$$\pi(a|s) = \frac{e^{Q(s,a)/\tau}}{\sum_{b \in \mathcal{A}} e^{Q(s,b)/\tau}}$$

Consider 4 actions with the following Q-values:

$$Q(s, a_1) = 2.0 \qquad Q(s, a_3) = 0.5$$
$$Q(s, a_2) = 1.0 \qquad Q(s, a_4) = -1.0$$

Softmax with temperature $\tau = 0.5$ (Low Temperature)

Step 1: Compute exponentials

$$e^{2.0/0.5} = e^{4.0} = 54.60 \qquad e^{0.5/0.5} = e^{1.0} = 2.72$$
$$e^{1.0/0.5} = e^{2.0} = 7.39 \qquad e^{-1.0/0.5} = e^{-2.0} = 0.14$$

Step 2: Sum of exponentials

$$\sum_{b \in \mathcal{A}} e^{Q(s,b)/\tau} = 54.60 + 7.39 + 2.72 + 0.14 = 64.85$$

Boltzmann Exploration

- Apply Soft-max function over the Q values (with temperature)

Consider 4 actions with the following Q-values:

$$Q(s,a_1) = 2.0 \qquad Q(s,a_3) = 0.5$$
$$Q(s,a_2) = 1.0 \qquad Q(s,a_4) = -1.0$$

$$\pi(a|s) = \frac{e^{Q(s,a)/\tau}}{\sum_{b\in\mathcal{A}} e^{Q(s,b)/\tau}}$$

Softmax with temperature $\tau = 0.5$ (Low Temperature)

Step 1: Compute exponentials

$$e^{2.0/0.5} = e^{4.0} = 54.60 \qquad e^{0.5/0.5} = e^{1.0} = 2.72$$
$$e^{1.0/0.5} = e^{2.0} = 7.39 \qquad e^{-1.0/0.5} = e^{-2.0} = 0.14$$

Step 2: Sum of exponentials

$$\sum_{b\in\mathcal{A}} e^{Q(s,b)/\tau} = 54.60 + 7.39 + 2.72 + 0.14 = 64.85$$

Step 3: Compute probabilities

$$\pi(a_1|s) = \frac{54.60}{64.85} = 0.84 \ (84\%) \qquad \pi(a_3|s) = \frac{2.72}{64.85} = 0.04 \ (4\%)$$
$$\pi(a_2|s) = \frac{7.39}{64.85} = 0.11 \ (11\%) \qquad \pi(a_4|s) = \frac{0.14}{64.85} = 0.00 \ (0\%)$$

Probabilities mass concentrates on the highest Q !!

Boltzmann Exploration

- Apply Soft-max function over the Q values (with temperature)

$$\pi(a|s) = \frac{e^{Q(s,a)/\tau}}{\sum_{b \in \mathcal{A}} e^{Q(s,b)/\tau}}$$

Consider 4 actions with the following Q-values:

$$Q(s, a_1) = 2.0 \qquad Q(s, a_3) = 0.5$$
$$Q(s, a_2) = 1.0 \qquad Q(s, a_4) = -1.0$$

Softmax with temperature $\tau = 2.0$ (High Temperature)

Step 1: Compute exponentials

$$e^{2.0/2.0} = e^{1.0} = 2.72 \qquad e^{0.5/2.0} = e^{0.25} = 1.28$$
$$e^{1.0/2.0} = e^{0.5} = 1.65 \qquad e^{-1.0/2.0} = e^{-0.5} = 0.61$$

Step 2: Sum of exponentials

$$\sum_{b \in \mathcal{A}} e^{Q(s,b)/\tau} = 2.72 + 1.65 + 1.28 + 0.61 = 6.26$$

Boltzmann Exploration

- Apply Soft-max function over the Q values (with temperature)

$$\pi(a|s) = \frac{e^{Q(s,a)/\tau}}{\sum_{b \in \mathcal{A}} e^{Q(s,b)/\tau}}$$

Consider 4 actions with the following Q-values:

$$Q(s, a_1) = 2.0 \qquad Q(s, a_3) = 0.5$$
$$Q(s, a_2) = 1.0 \qquad Q(s, a_4) = -1.0$$

Softmax with temperature $\tau = 2.0$ (High Temperature)

Step 1: Compute exponentials

$$e^{2.0/2.0} = e^{1.0} = 2.72 \qquad e^{0.5/2.0} = e^{0.25} = 1.28$$
$$e^{1.0/2.0} = e^{0.5} = 1.65 \qquad e^{-1.0/2.0} = e^{-0.5} = 0.61$$

Step 2: Sum of exponentials

$$\sum_{b \in \mathcal{A}} e^{Q(s,b)/\tau} = 2.72 + 1.65 + 1.28 + 0.61 = 6.26$$

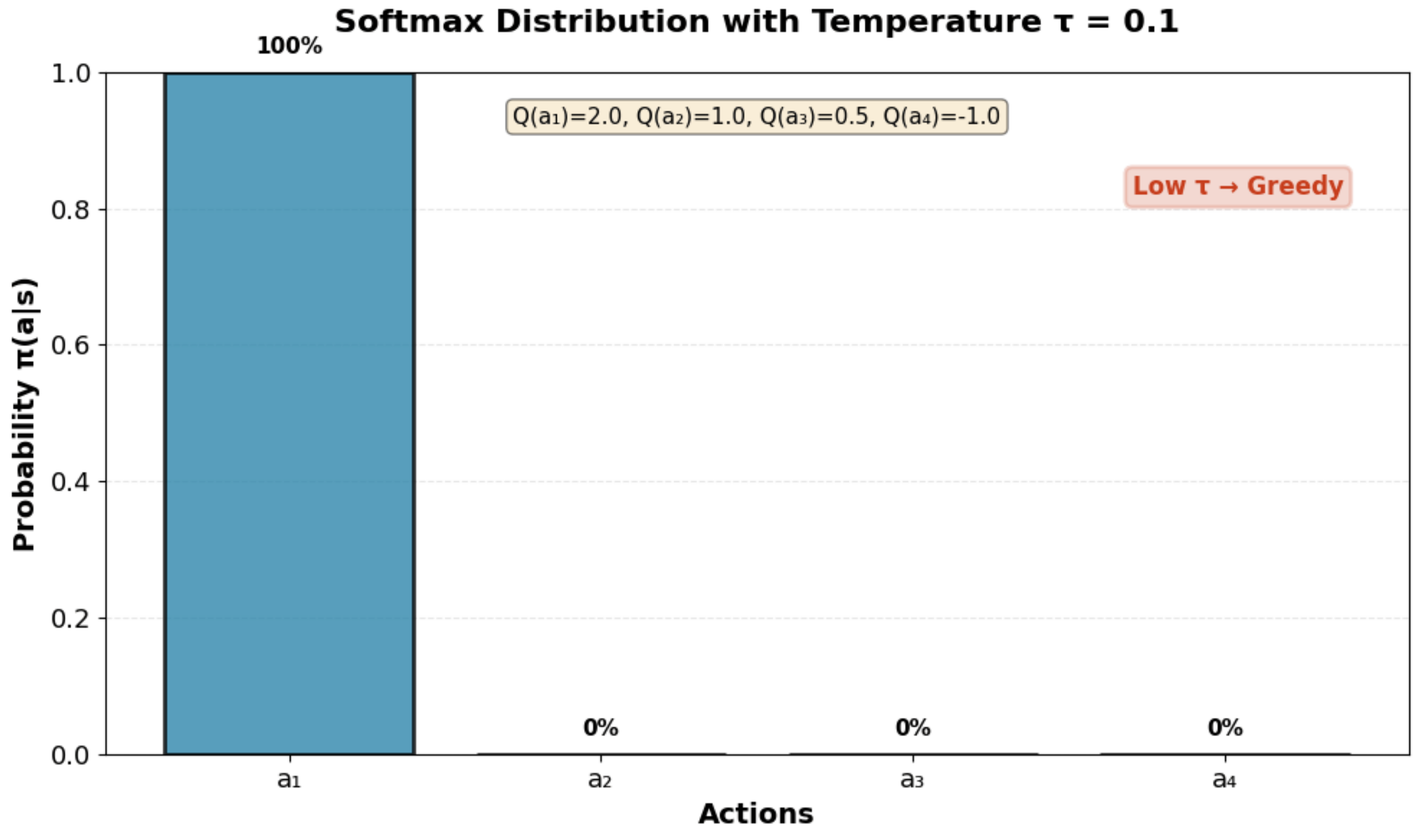
Step 3: Compute probabilities

$$\pi(a_1|s) = \frac{2.72}{6.26} = 0.43 \ (43\%) \qquad \pi(a_3|s) = \frac{1.28}{6.26} = 0.20 \ (20\%)$$
$$\pi(a_2|s) = \frac{1.65}{6.26} = 0.26 \ (26\%) \qquad \pi(a_4|s) = \frac{0.61}{6.26} = 0.10 \ (10\%)$$

Probabilities mass becomes more uniform !!

Boltzmann Exploration

- Apply Soft-max function over the Q values (with temperature)
- Lower τ favors action with higher Q and higher τ approaches uniform

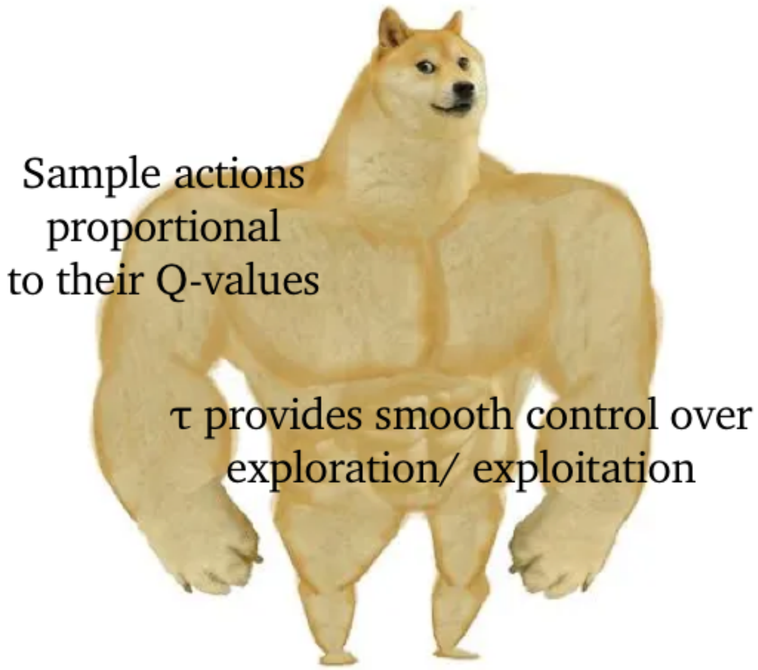


Boltzmann Exploration

- Is this better than ε-greedy action selection?

Boltzmann Exploration

- Is this better than ε-greedy action selection?
  - ✦ Not strictly better, just more sophisticated



Boltzmann Exploration

ε-greedy Exploration

Upper Confidence Bound

Upper Confidence Bound

- Select action based on their potential of being optimal
- The equation takes this form:

$$A_t \doteq \arg\max_a \left[\qquad \right]$$

Upper Confidence Bound

$$A_t \doteq \arg\max_a \left[\quad \right]$$

Whatever term exists here, evaluate it over values of `a` and select the `a` that results in the maximum

Upper Confidence Bound

$$A_t \doteq \arg\max_a \left[Q_t(a) \right]$$

Value estimate

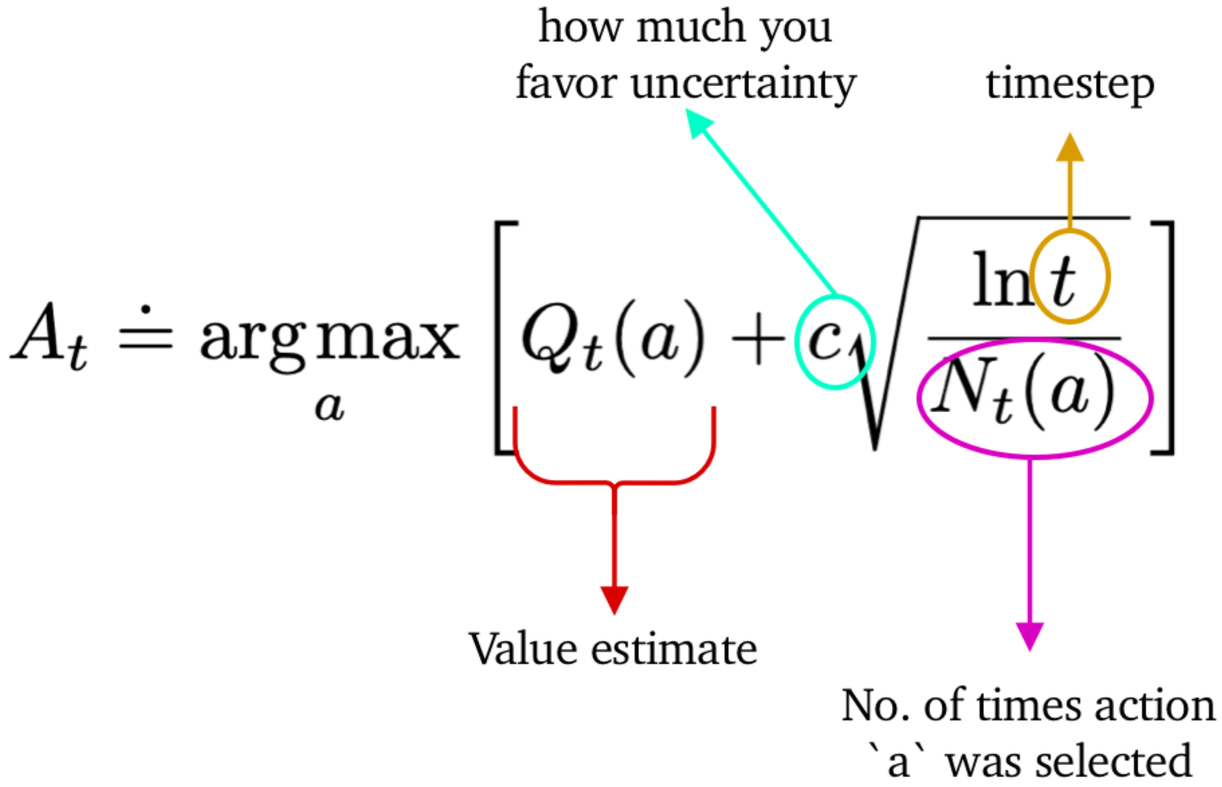
Upper Confidence Bound

$$A_t \doteq \arg\max_a \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

Value estimate

Variance in the Value estimate
(does not include `c`)

Upper Confidence Bound



Upper Confidence Bound

- Worked out example:
  - ✦ Two actions A and B
  - ✦ t = 10 and c = 1

A : Q = 1.0, N = 8

B : Q = 0.8, N = 2

Upper Confidence Bound

- Worked out example:
  - ✦ Two actions A and B
  - ✦ t = 10 and c = 1

A : Q = 1.0, N = 8, UCB ≈ 1 + 0.53 = 1.53

B : Q = 0.8, N = 2, UCB ≈ 0.8 + 1.07 = 1.87

- Select B although B's value estimate is lower! Why?

Upper Confidence Bound

- Works well for the Bandit problem
- AlphaGo combines Monte Carlo Tree Search with UCB*

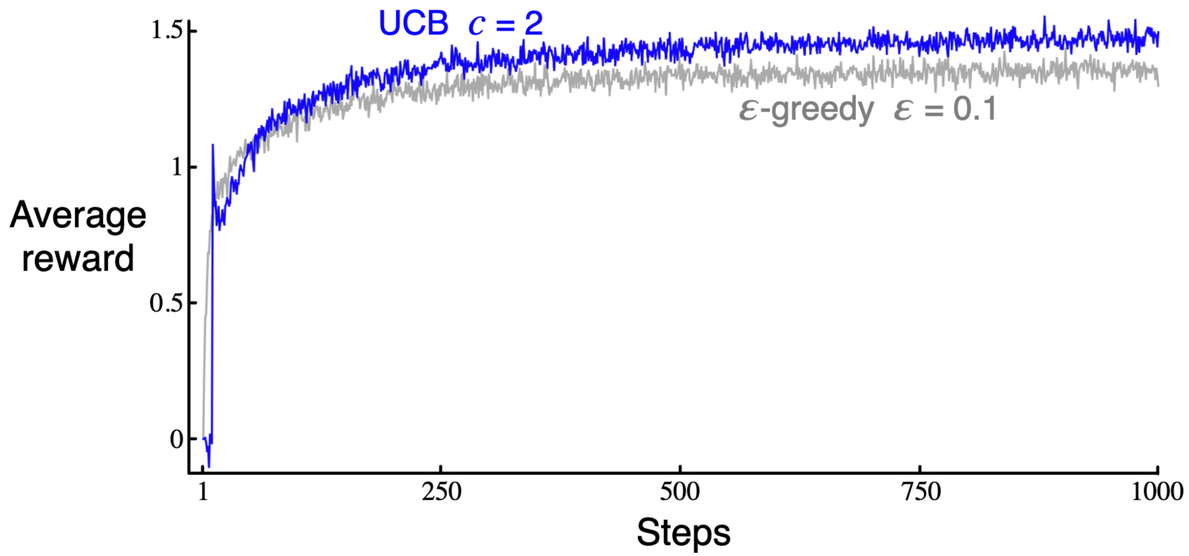


Figure 2.4: Average performance of UCB action selection on the 10-armed testbed. As shown, UCB generally performs better than ε-greedy action selection, except in the first k steps, when it selects randomly among the as-yet-untried actions.

Sutton & Barto, Reinforcement Learning, Second edition.

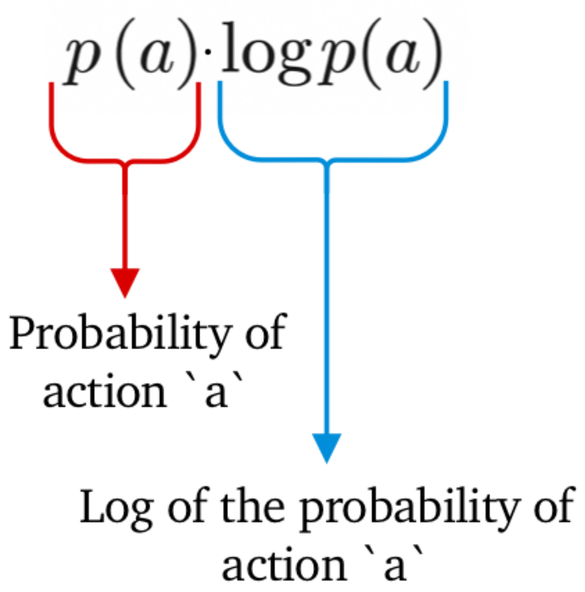
Entropy Regularization

Entropy

- Amount of uncertainty or surprise (information theory)
- Higher entropy ⟶ more surprise

Entropy

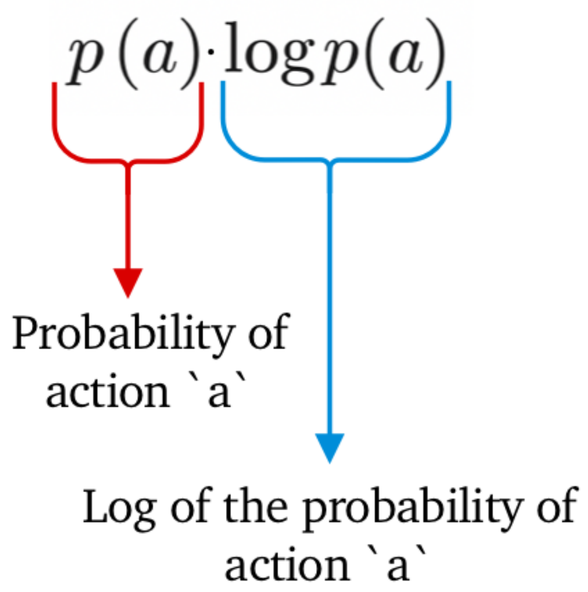
- “Weighted surprise”, worked out example



Entropy

- “Weighted surprise”, worked out example

- **Event A** with p = 0.8 (highly likely)
  - ✦ log(p) = −0.223
  - ✦ p × log(p) = 0.8 × (−0.223) = **−0.179**

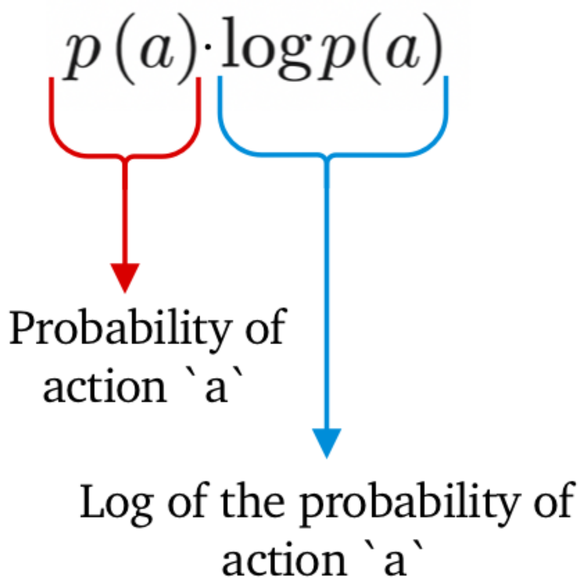


Entropy

- “Weighted surprise”, worked out example

- **Event A** with p = 0.8 (highly likely)
  - ✦ log(p) = −0.223
  - ✦ p × log(p) = 0.8 × (−0.223) = **−0.179**

- **Event B** with p = 0.1 (rare)
  - ✦ log(p) = −2.302
  - ✦ p × log(p) = 0.1 × (−2.302) = **−0.230**



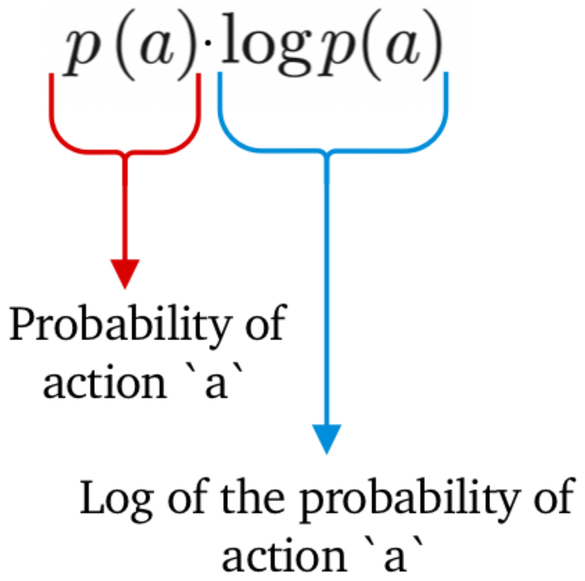
Entropy

- “Weighted surprise”, worked out example

- **Event A** with p = 0.8 (highly likely)
  - log(p) = −0.223
  - p × log(p) = 0.8 × (−0.223) = **−0.179**

- **Event B** with p = 0.1 (rare)
  - log(p) = −2.302
  - p × log(p) = 0.1 × (−2.302) = **−0.230**

- **Event C** with p = 0.36 (peak uncertainty)
  - log(p) = −0.999
  - p × log(p) = 0.36 × (−0.999) = **−0.368**



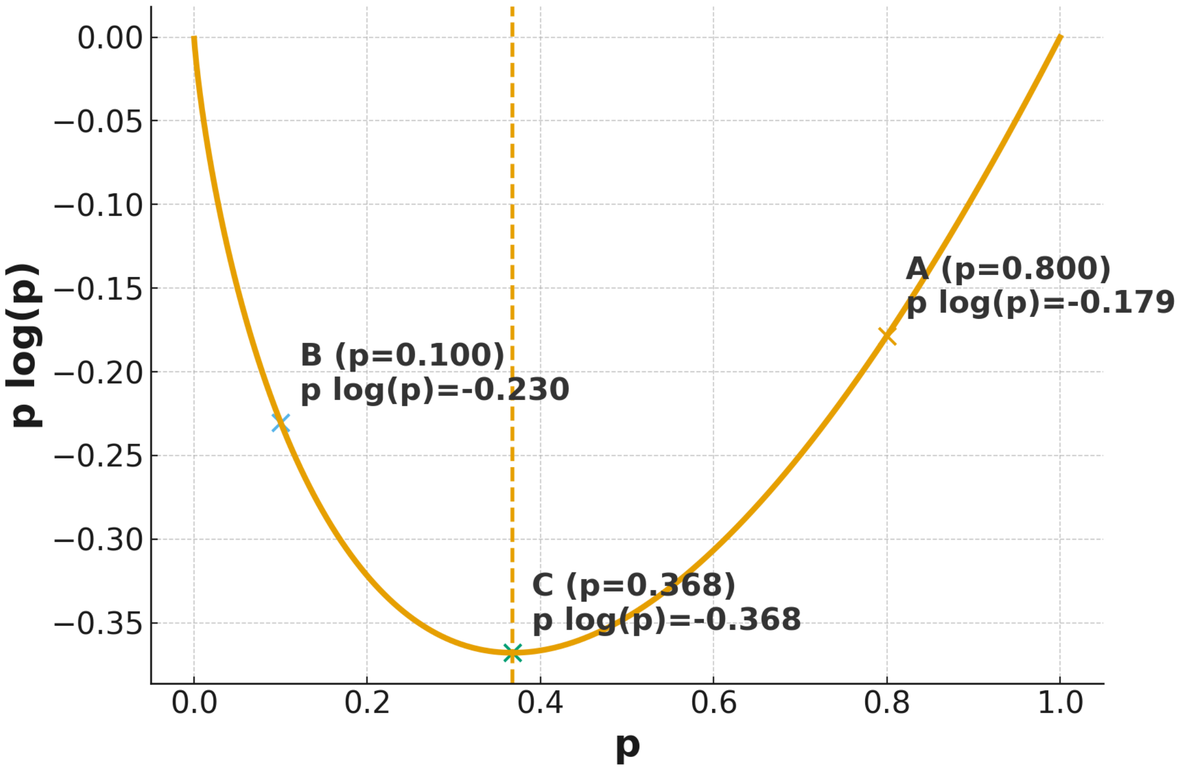
Entropy

- “Weighted surprise”, worked out example

- **Event A** with p = 0.8 (highly likely)
  - log(p) = −0.223
  - p × log(p) = 0.8 × (−0.223) = **−0.179**

- **Event B** with p = 0.1 (rare)
  - log(p) = −2.302
  - p × log(p) = 0.1 × (−2.302) = **−0.230**

- **Event C** with p = 0.36 (peak uncertainty)
  - log(p) = −0.999
  - p × log(p) = 0.36 × (−0.999) = **−0.368**

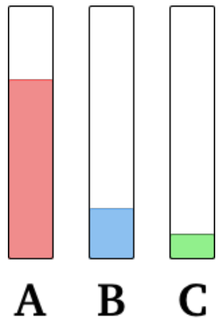


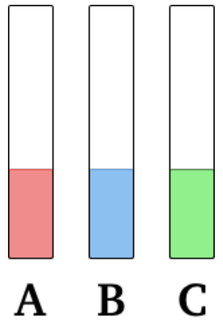
Entropy

$$H = -\sum_{a} p(a) \log p(a)$$

Entropy Regularization

$$H = -\sum_{a} p(a) \log p(a)$$

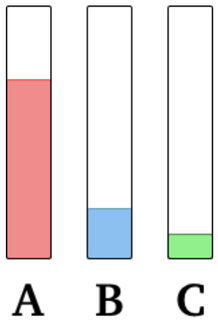




Entropy Regularization

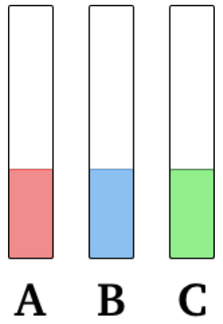
$$H = -\sum_{a} p(a) \log p(a)$$

Low entropy



$$H = -\big(0.7 \log_2 0.7 + 0.2 \log_2 0.2 + 0.1 \log_2 0.1\big)$$
$$= -\big(-0.2497 - 0.3219 - 0.2303\big)$$
$$H = 0.802$$

High entropy



$$H = -\sum_{i=1}^{3} \frac{1}{3} \log_2 \frac{1}{3} = -\log_2 \frac{1}{3} = \log_2 3$$
$$H = 1.099$$

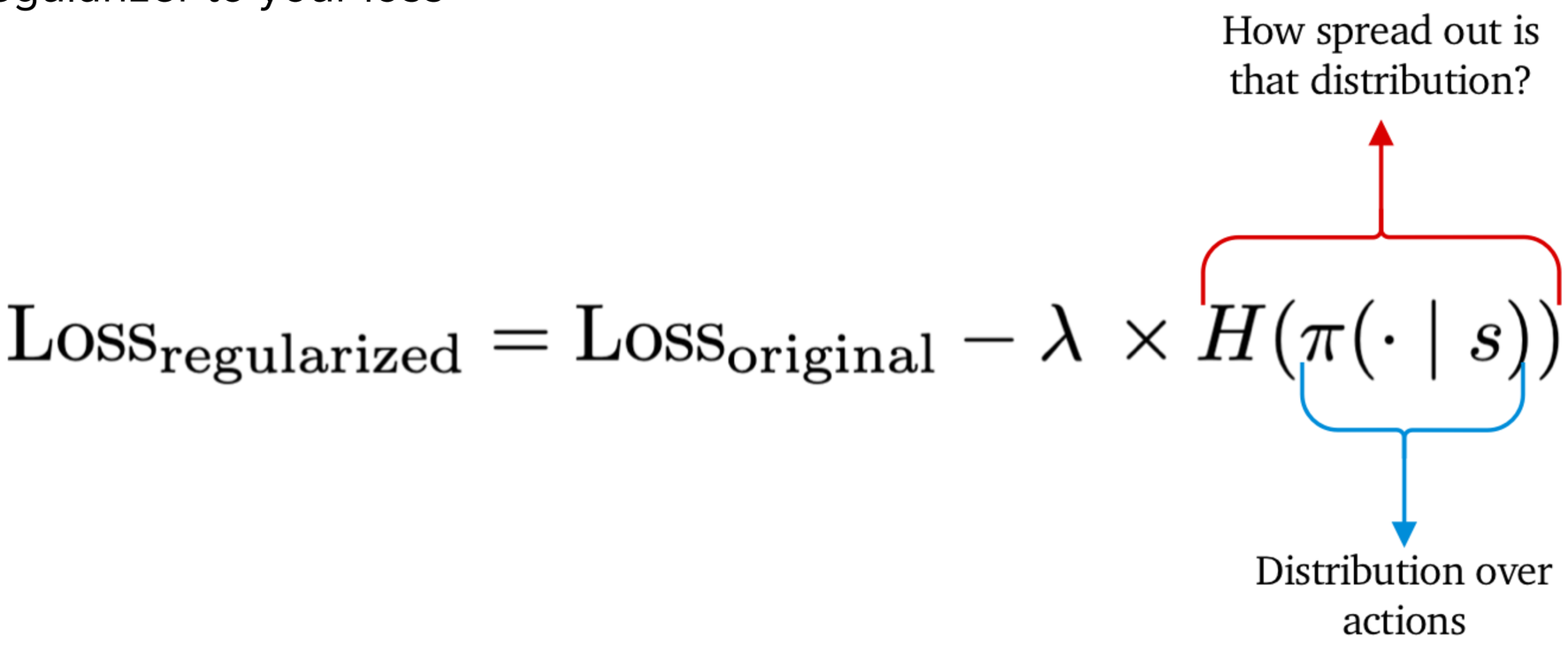
Entropy Regularization

- Add a regularizer to your loss

$$\text{Loss}_{\text{regularized}} = \text{Loss}_{\text{original}} - \lambda \times H(\pi(\cdot \mid s))$$

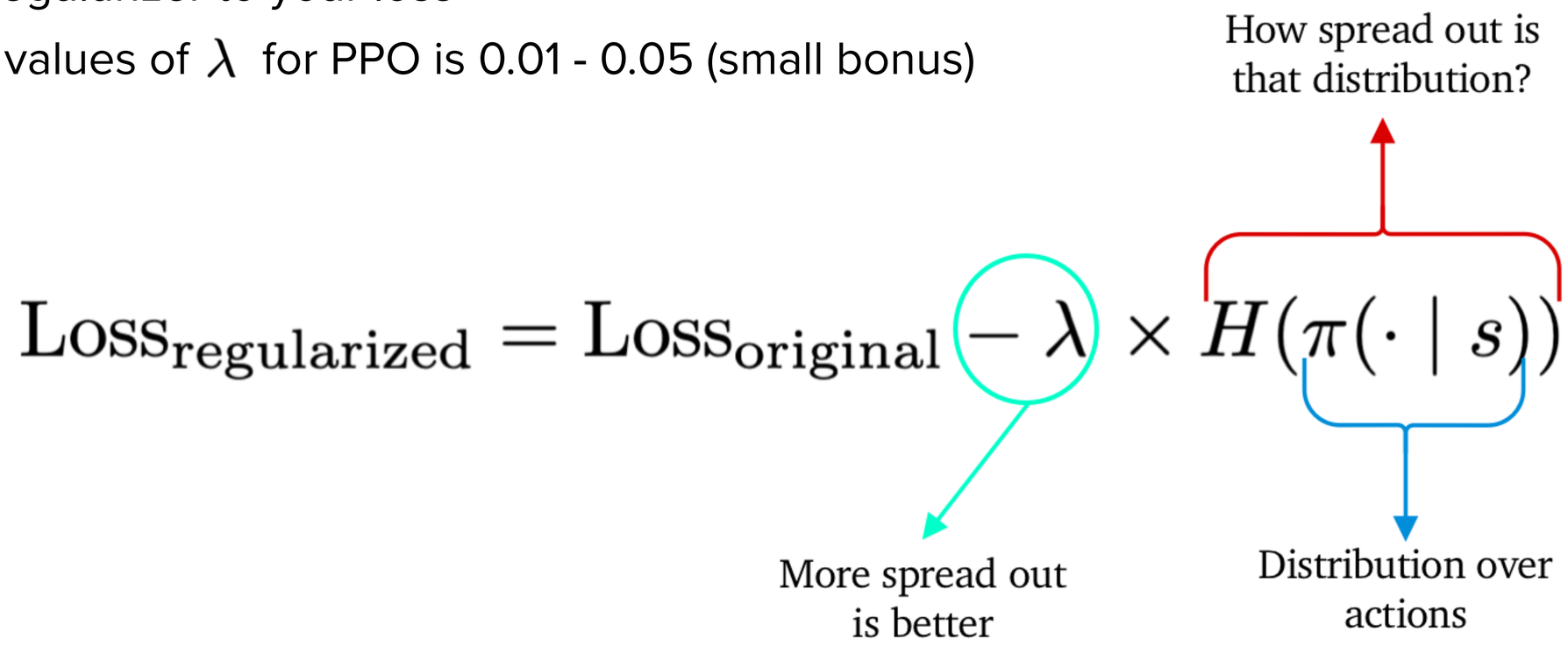
Entropy Regularization

- Add a regularizer to your loss



Entropy Regularization

- Add a regularizer to your loss
- Typical values of λ for PPO is 0.01 - 0.05 (small bonus)



Random Network Distillation

Random Network Distillation

- Montezuma's Revenge

Couldn't even find the first reward !!

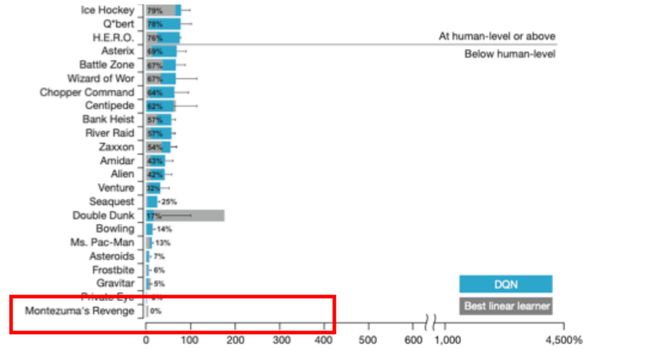


Figure 3 | Comparison of the DQN agent with the best reinforcement learning methods[15] in the literature. The performance of DQN is normalized outperforms competing methods (also see Extended Data Table 2) the games, and performs at a level that is broadly comparable with

DQN

frostbite	496.1	190.5	1448.1	2,930.2	2,332.4	2,813.9	418.8	**4,141.1**
gopher	8190.4	10022.8	15253.0	57,783.8	20,051.4	27,778.3	13,131.0	**72,595.7**
gravitar	298.0	303.5	200.5	218.0	297.0	422.0	250.5	**567.5**
hero	14992.9	32464.1	14892.5	20,506.4	15,207.9	28,554.2	2,454.2	**50,496.8**
ice_hockey	-1.6	-2.8	-2.5	-1.0	-1.3	**-0.1**	-2.4	-0.7
kangaroo	4496.0	94.0	**11204.0**	10,241.0	10,334.0	9,555.5	7,465.0	10,841.0
krull	6206.0	5560.0	6796.1	7,406.5	**8,051.6**	6,757.8	6,833.5	6,715.5
kung_fu_master	20882.0	28819.0	30207.0	31,244.0	24,288.0	**33,896.0**	27,921.0	28,999.8
montezuma_revenge	47.0	67.0	42.0	13.0	22.0	130.0	55.0	**154.0**
ms_pacman	1092.3	655.7	1241.3	1,824.6	2,250.6	2,064.1	1,012.1	**2,570.2**
name_this_game	6738.8	10476.1	8960.3	**11,836.1**	11,185.1	11,382.3	7,186.4	11,686.5
phoenix	7484.8	52894.1	12366.5	27,430.1	20,410.5	31,358.3	15,505.0	**103,061.6**
pitfall	-113.2	-78.5	-186.7	**-14.8**	-46.9	-342.8	-154.4	-37.6
pong	18.0	5.6	**19.1**	18.9	18.8	18.9	18.0	19.0
private_eye	207.9	206.9	575.5	179.0	292.6	5,717.5	**5,955.4**	1,704.4

Rainbow

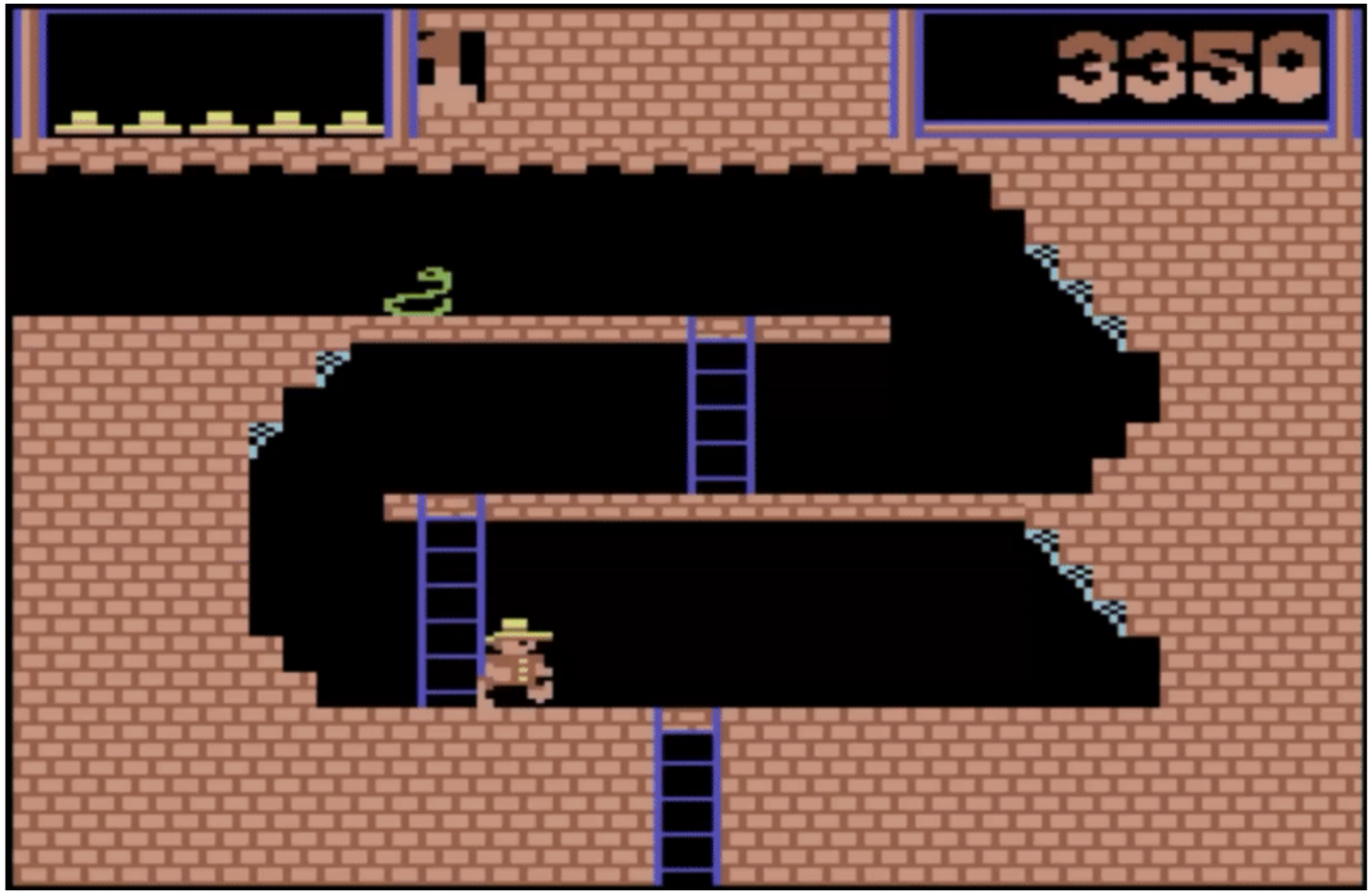
Gravitar	194.0	225.3	737.2
IceHockey	-6.4	-5.9	**-4.2**
Jamesbond	52.3	261.8	**560.7**
Kangaroo	45.3	50.0	**9928.7**
Krull	**8367.4**	7268.4	7942.3
KungFuMaster	24900.3	**27599.3**	23310.3
MontezumaRevenge	0.0	0.3	**42.0**
MsPacman	1626.9	**2718.5**	2096.5
NameThisGame	5961.2	**8488.0**	6254.9
Pitfall	-55.0	**-16.9**	-32.9
Pong	19.7	**20.7**	20.7

PPO

DQN

Random Network Distillation

- Montezuma’s Revenge

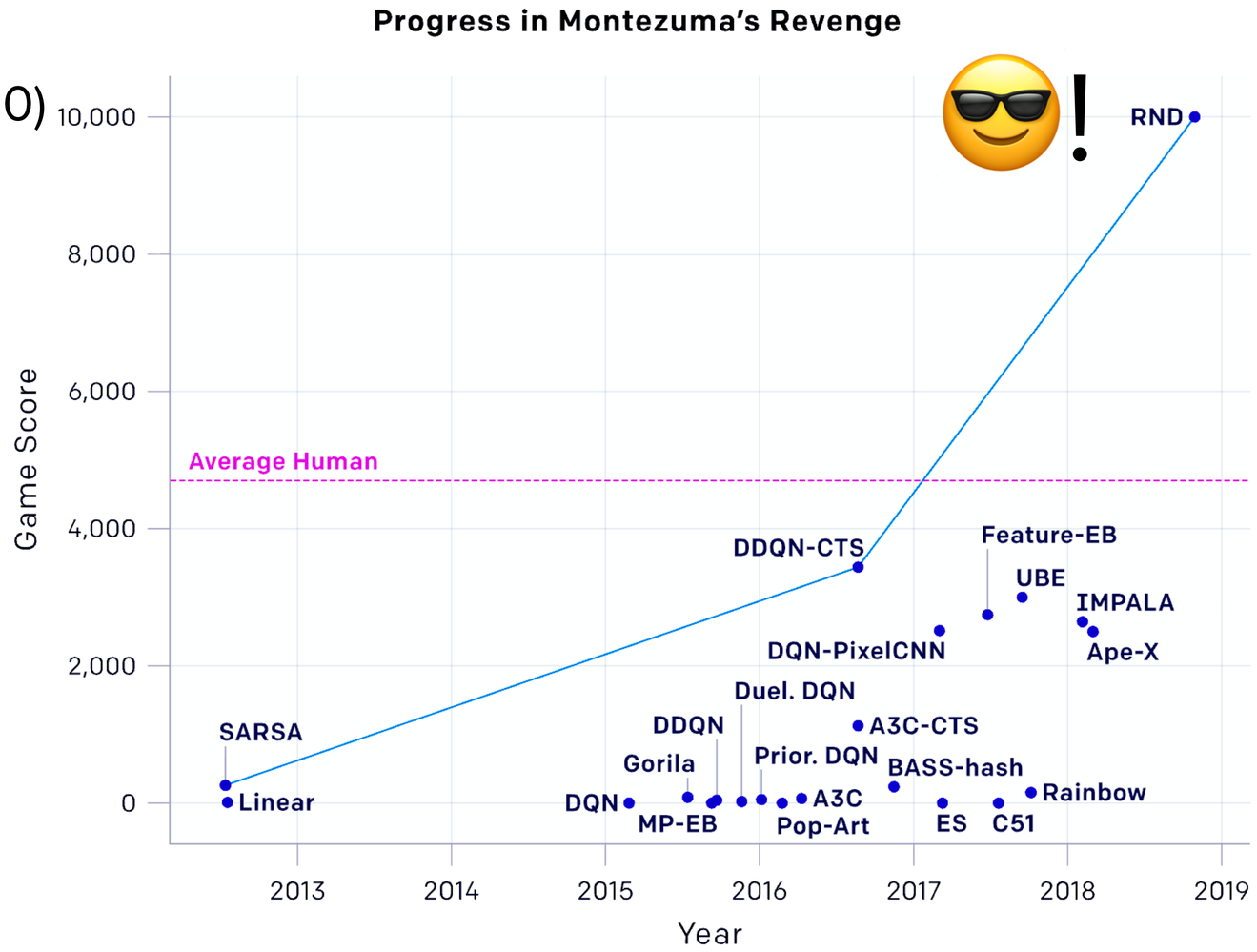


Random Network Distillation

- Montezuma’s Revenge
  - ✦ Subgoals: grab a key now, unlock a door later
  - ✦ Long action sequences
  - ✦ Detours

Random Network Distillation

- Montezuma's Revenge
  - ✦ **~10,000 points** (avg. human: ~4,700)
  - ✦ **Major breakthrough!**



https://openai.com/index/reinforcement-learning-with-prediction-based-rewards/

Random Network Distillation

- Intrinsic reward
  - ✦ Get it from the environment → Extrinsic reward
  - ✦ Generated by the agent itself → Intrinsic reward

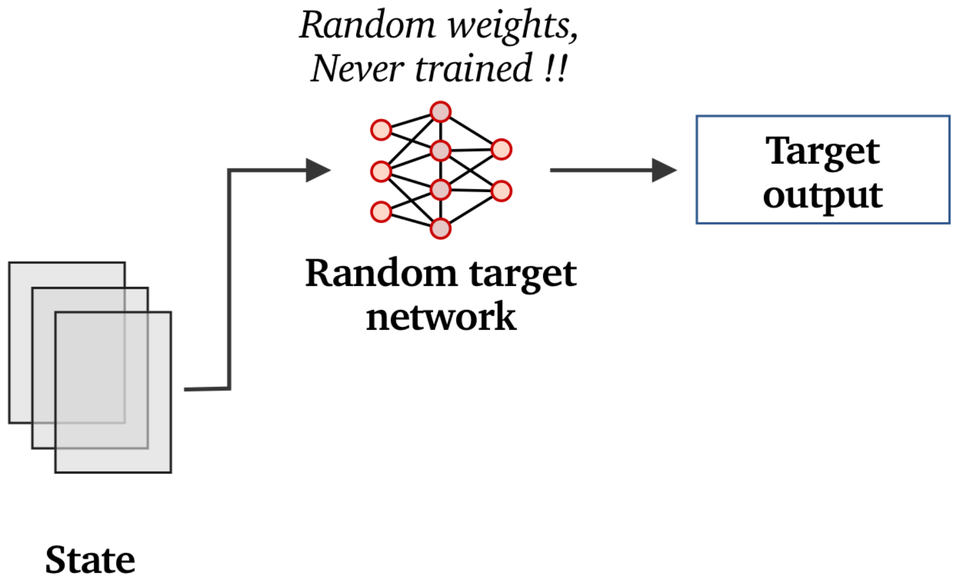
Random Network Distillation

- Intrinsic reward
  - ✦ Get it from the environment → Extrinsic reward
  - ✦ Generated by the agent itself → Intrinsic reward

- Intrinsic reward mechanism in RND
  - ✦ “Explore un-familiar states”

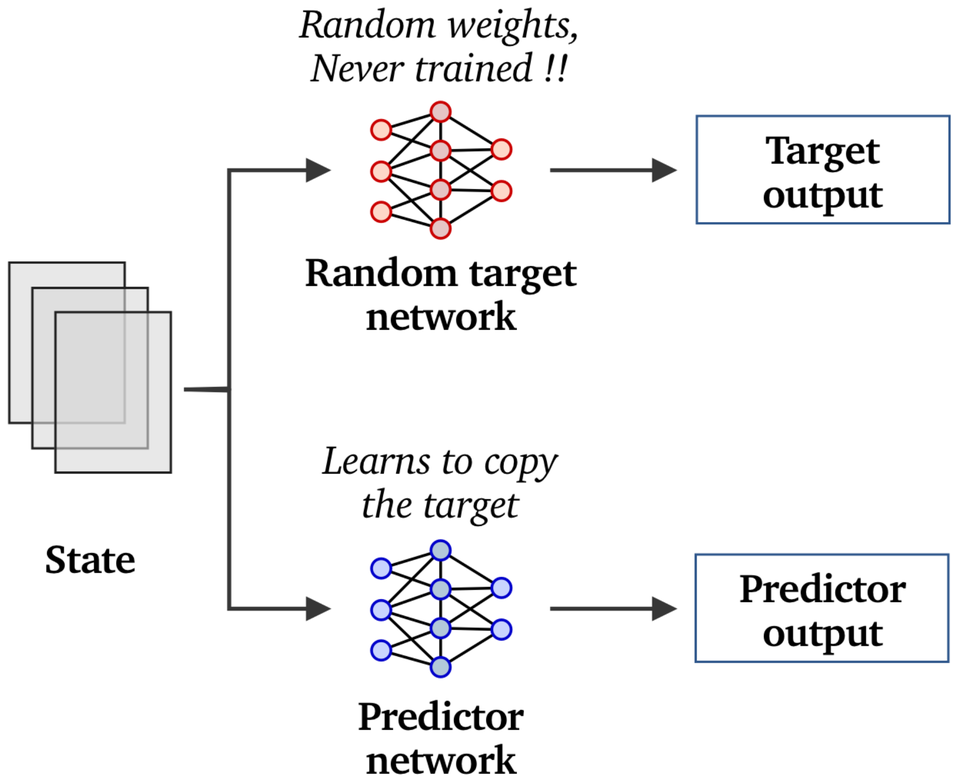
Random Network Distillation

- Intrinsic reward mechanism in RND



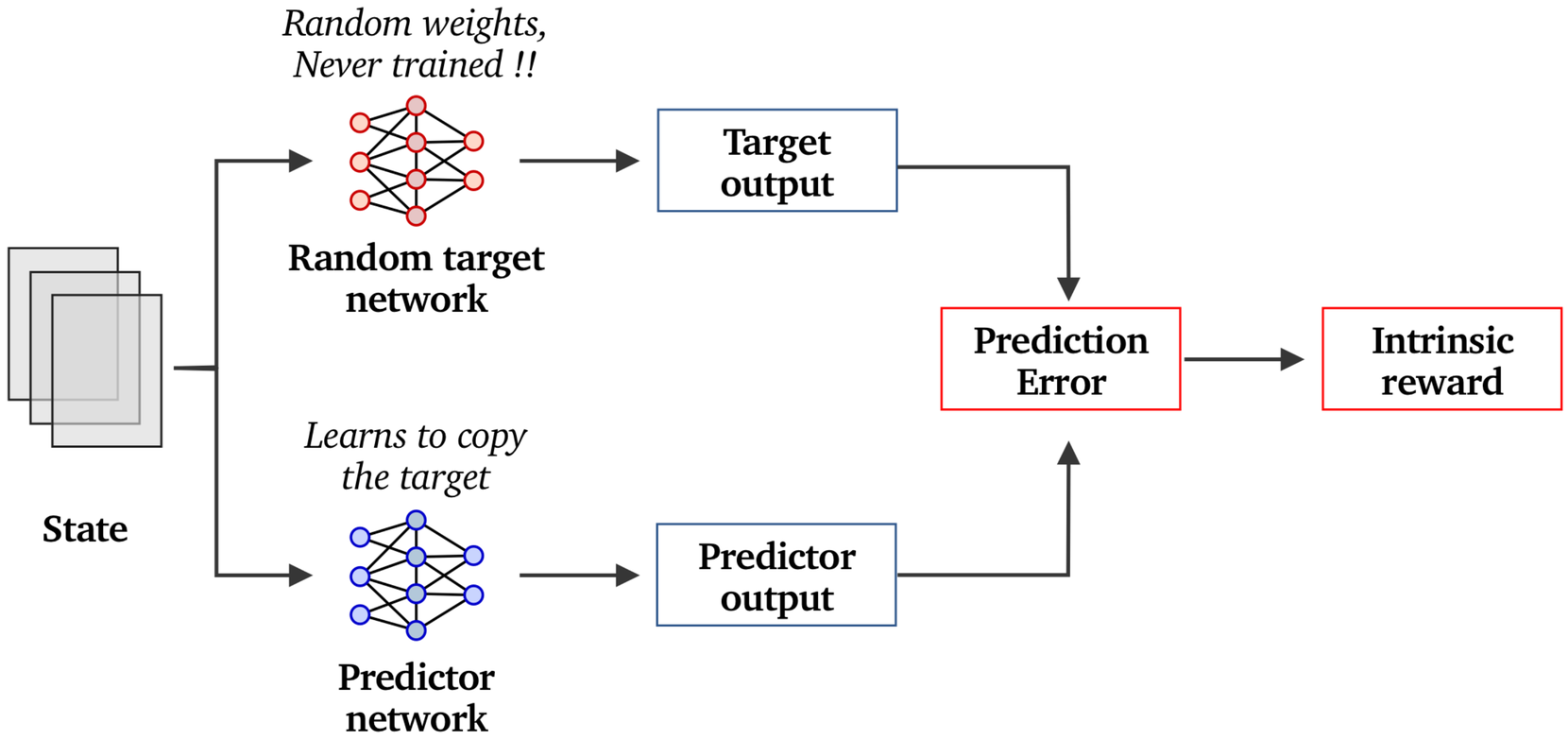
Random Network Distillation

- Intrinsic reward mechanism in RND



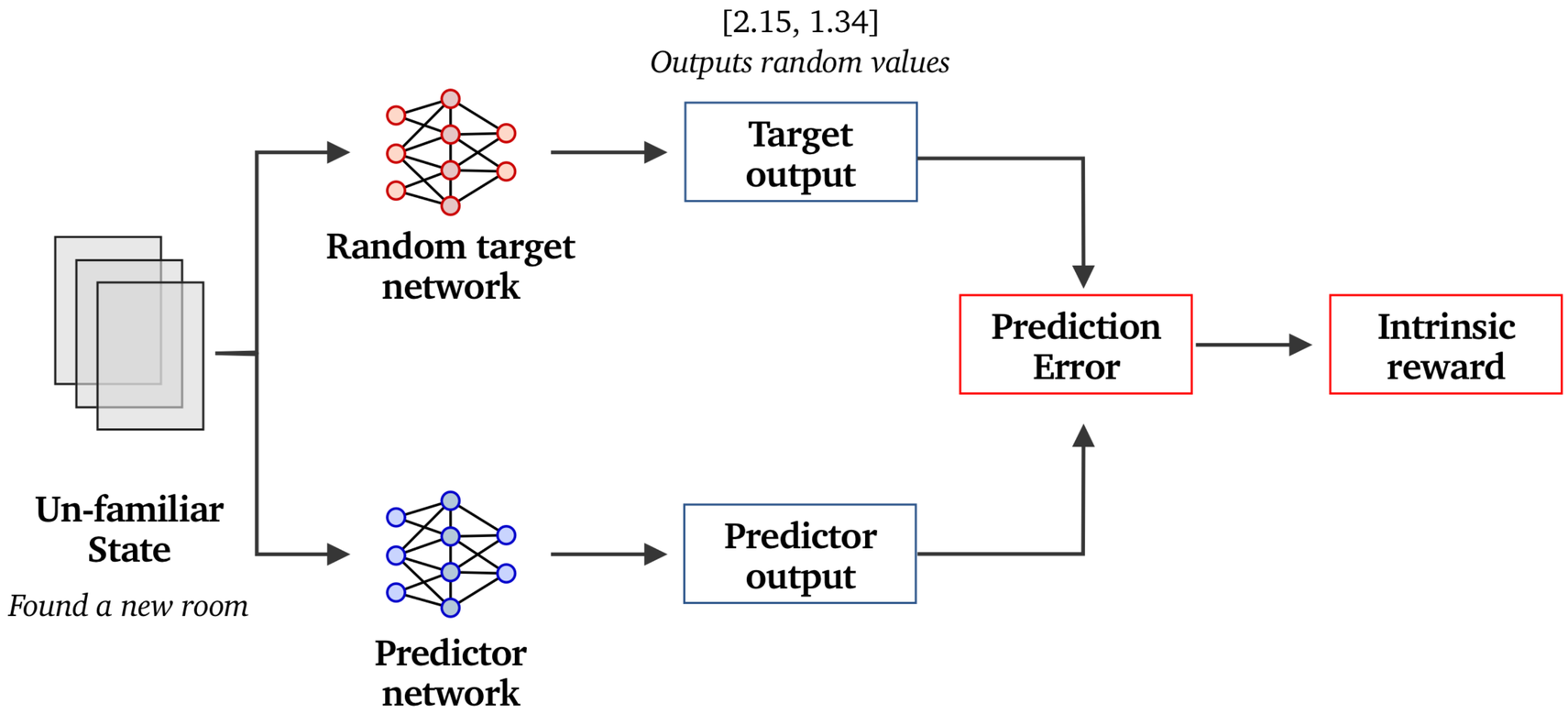
Random Network Distillation

- Intrinsic reward mechanism in RND



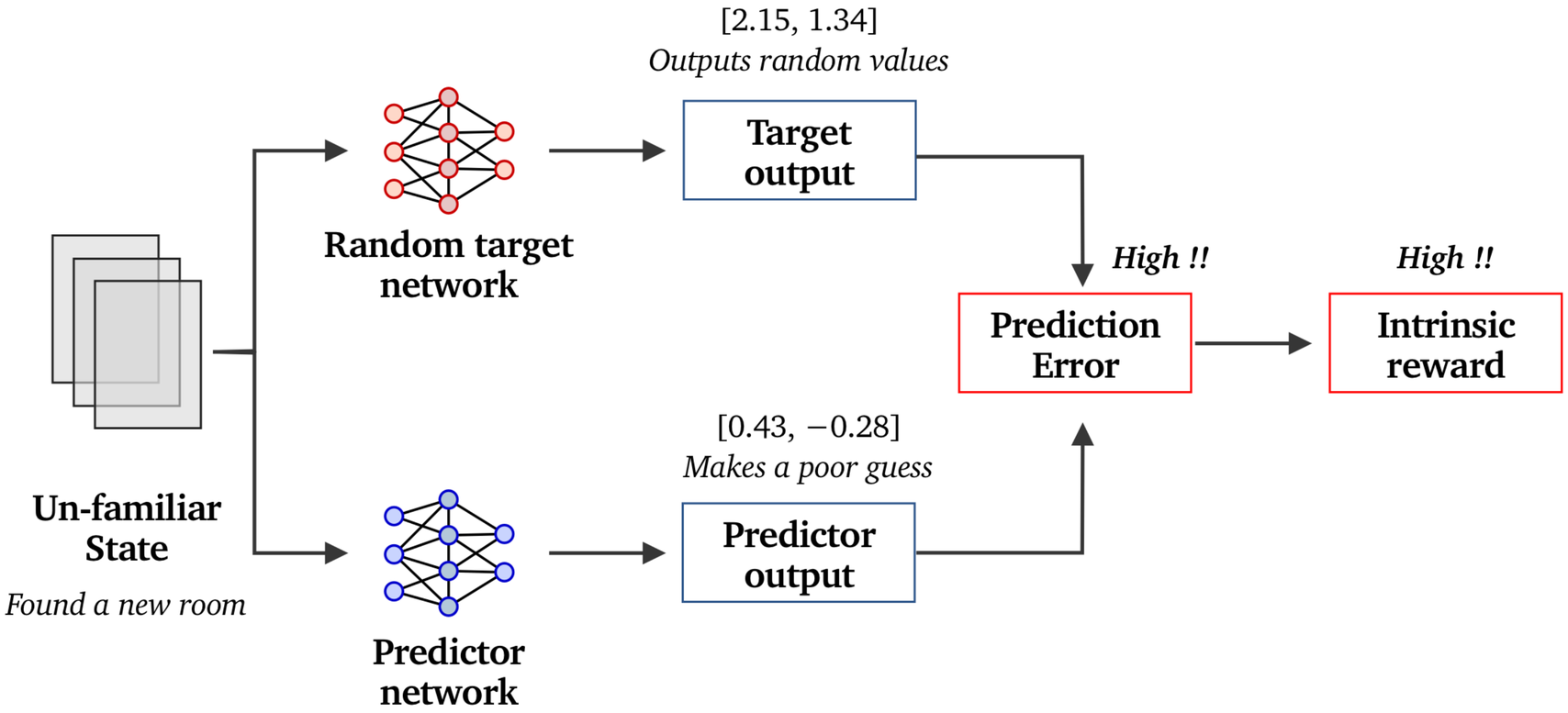
Random Network Distillation

- Intrinsic reward mechanism in RND



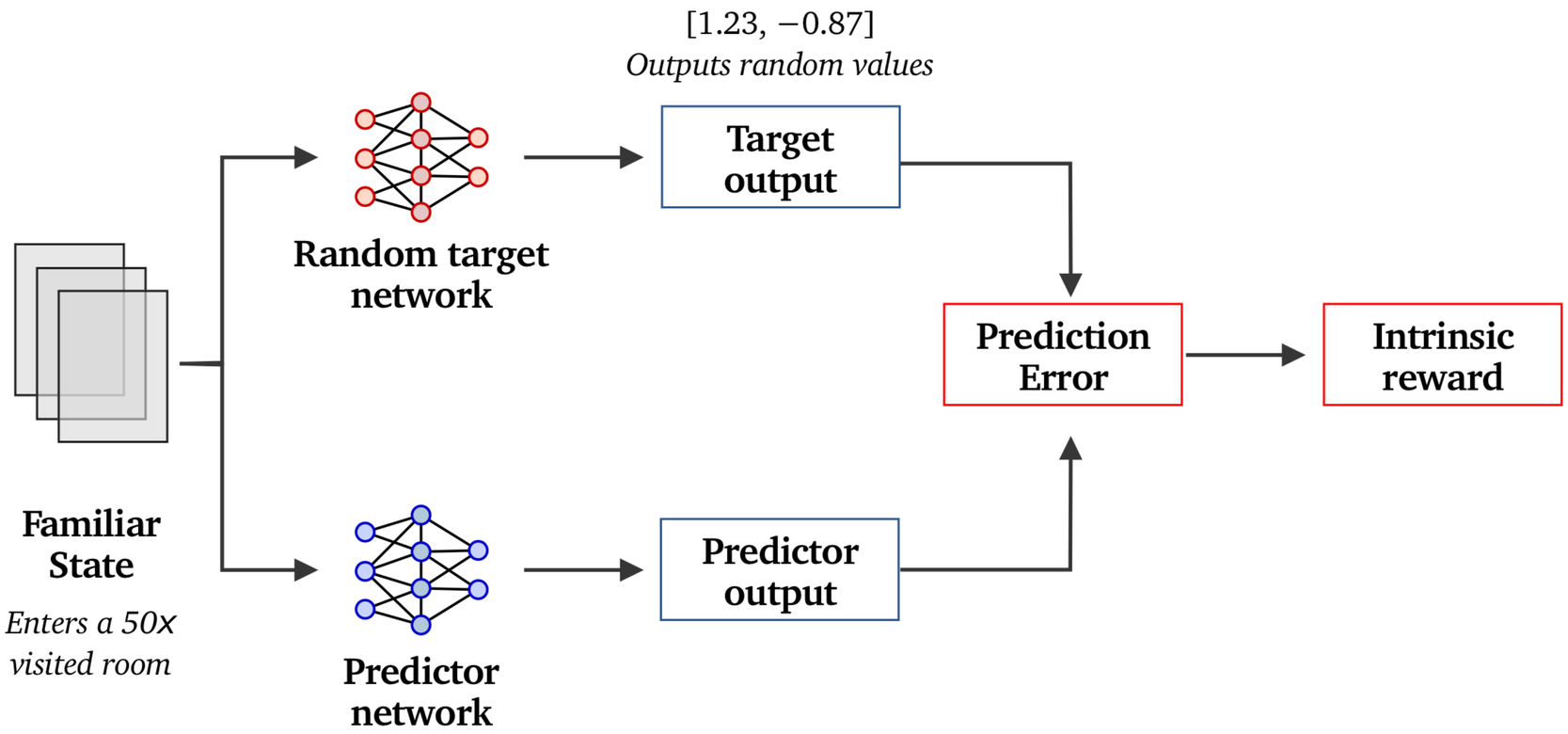
Random Network Distillation

- Intrinsic reward mechanism in RND



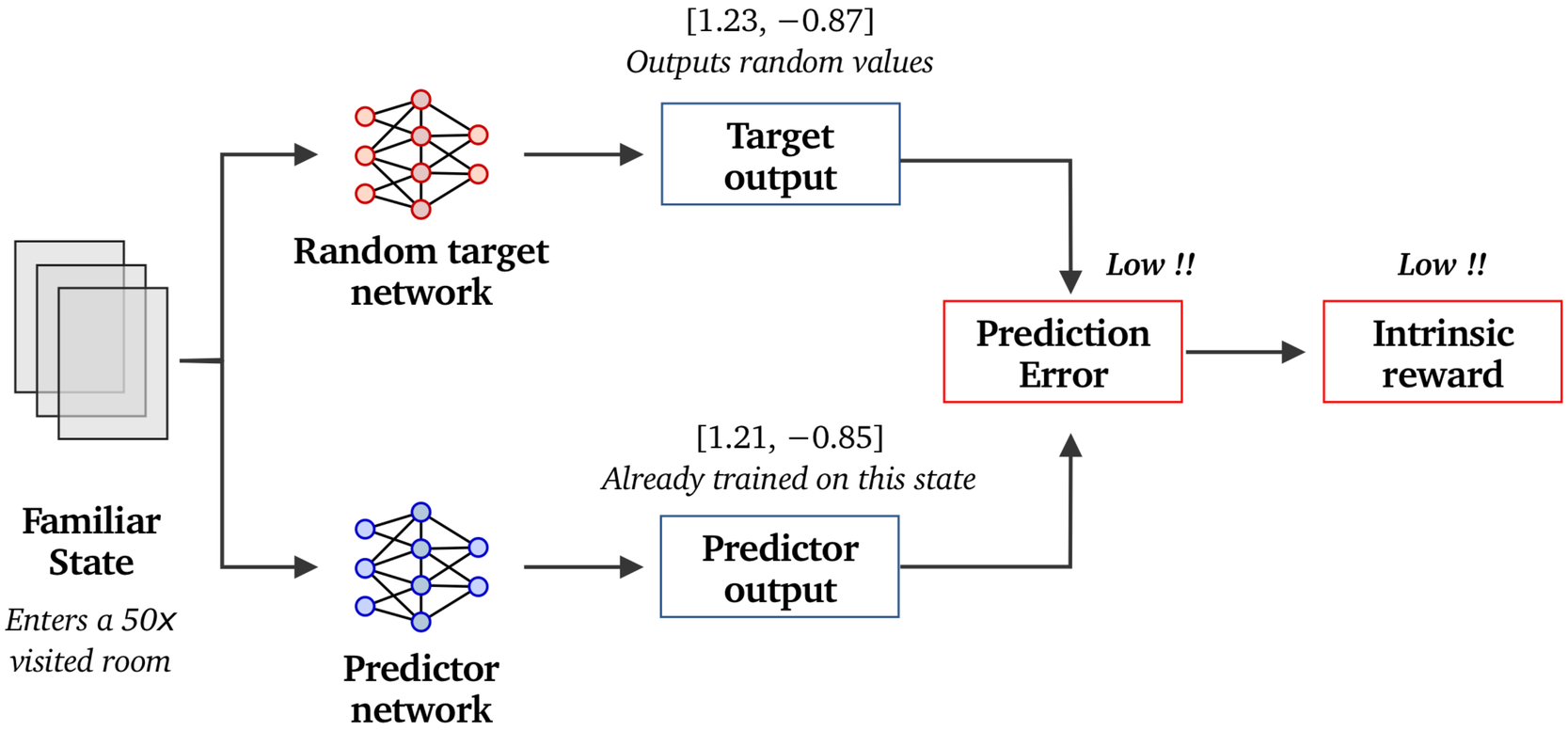
Random Network Distillation

- Intrinsic reward mechanism in RND



Random Network Distillation

- Intrinsic reward mechanism in RND



Go-Explore

- Further improvement on Montezuma’s revenge: Go-Explore
- Problems with previous algorithms
  - ✦ **Detachment:** "forgets" how to get back to promising states it visited before
  - ✦ **Derailment:** tries to return to a promising state, but stochastic actions or exploration noise knock it off course

Go-Explore

- Further improvement on Montezuma’s revenge: Go-Explore
- Problems with previous algorithms
  - ✦ **Detachment:** "forgets" how to get back to promising states it visited before
  - ✦ **Derailment:** tries to return to a promising state, but stochastic actions or exploration noise knock it off course

- Solution: Keep a memory of discovered states
- More systematic exploration
- Score: ~**2,000,000**+ points (superhuman!) 😎!

Summary

- Random exploration: ε-greedy
- Value weighted: Boltzmann
- Uncertainty driven: UCB
- Surprise driven: Entropy regularization
- Intrinsic reward mechanism: RND
- Memory-based: Go-Explore

Thank You!