Exploration

Hao Su

(slides prepared in tandem with Quan Vuong, Tongzhou Mu and Zhiao Huang)

Spring, 2021

Some contents are based on ***Bandit Algorithms*** *from Dr. Tor Lattimore and Prof. Csaba Szepesvári, and* ***COMPM050/COMPGI13*** *taught at UCL by Prof. David Silver.*

Agenda

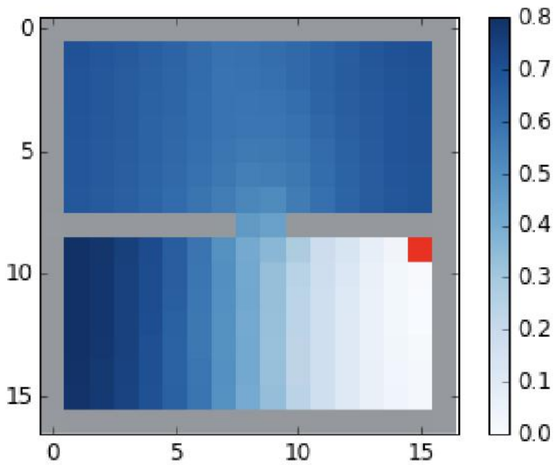
- Exploration versus Exploitation
- Multi-Armed Bandits
- Intrinsic Rewards

click to jump to the section.

Exploration versus Exploitation

Exploration is Difficult

- Curse of Dimensionality:
  - It is common for the state space to be high-dimensional (e.g., image input, multi-joint arms)
  - The volume of effective state space grows *exponentially* w.r.t. dimension!
- Even exploration in low-dimensional space may be tricky when there are "alleys":



The Laplacian in RL: Learning Representations with Efficient Approximations, Wu et al.

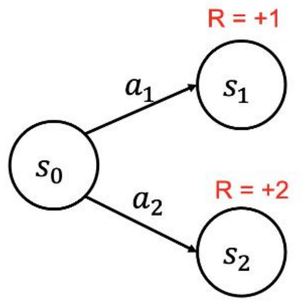
Balancing Exploration and Exploitation

- Goal: select actions to maximize expected return.
- Problem:
  - Actions may have long-term consequences.
  - Reward may be delayed.
- It may be better to sacrifice immediate reward to gain more long-term reward.
- A high performing policy trade-offs between exploration and exploitation:
  - Exploration: take action to learn more about the MDP.
  - Exploitation: take the action currently **believed** to maximize expected return.

Multi-Armed Bandits

Examples from Lecture 13

- We can remove the state space $\{s_0, s_1, s_2\}$ since action does not need to be conditioned on the state.
- The MDP is now defined by the action space $\{a_1, a_2\}$ and the reward function $\mathcal{R}$:
  - $\mathcal{R}(a_1) = 1$
  - $\mathcal{R}(a_2) = 2$
- Such one-step MDP has a special name "Multi-Armed Bandits".
- Very extensively studied and widely deployed in industry.



Multi-Armed Bandits

- A multi-armed bandit is a tuple of action space and reward function $(\mathcal{A}, \mathcal{R})$.
- $\mathcal{A}$ is a finite and known set of actions.
- $\mathcal{R}$ is a function mapping an action to an unknown probability distribution over rewards.
  - $\mathcal{R}(a) = \Pr[R|a]$
- At each step t, the agent selects an action a_t, the environment generates a reward $r_t \sim Pr[R|a_t]$.
  - Note that a_t can be realization of a random variable even if the policy is a deterministic function of input.
- Goal is to maximize cumulative reward $\sum_{t=1}^{T} r_t$

Regret

- Formal statement about the performance of a policy needs to be made relative to the optimal policy.
- We therefore introduce the notion of **regret**.

Regret

- The action-value of action a is its expected reward:

$$Q(a) = \mathbb{E}_R[R|a]$$

- The optimal value V^* is the action-value of the optimal action.

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

- The regret of a policy $A_t \sim \pi$ at time t is the gap between the optimal value and the expected action-value.

$$l_t = V^* - \mathbb{E}_{A_t}\left[Q(A_t)\right] = \mathbb{E}_{A_t}\left[V^* - Q(A_t)\right]$$

- The total regret up to time T is the sum of the timestep regret up to time T.

$$L_T = \sum_{t=1}^{T} \mathbb{E}_{A_t}\left[V^* - Q(A_t)\right] = \mathbb{E}_{A_t}\left[\sum_{t=1}^{T}[V^* - Q(A_t)]\right]$$

- The total regret is sometimes referred to as regret.

Total Regret Decomposition

- The count $N_T(a)$ is the number of times action a was chosen up to time T.

$$N_T(a) = \sum_{t=1}^{T} 1_{A_t=a}$$

- The gap Δ_a is the difference in value between action a and the optimal action a^*.

$$\Delta_a = V^* - Q(a) = V^* - \mathbb{E}_R[R|a]$$

- The total regret is a function of gaps and counts.

$$L_T = \mathbb{E}_{\forall A_t}\left[\sum_{t=1}^{T}[V^* - Q\left(A_t\right)]\right] = \mathbb{E}_{\forall A_t}\left[\sum_{t=1}^{T}\sum_{a\in\mathcal{A}} 1_{A_t=a}[V^* - Q\left(a\right)]\right] = \sum_{a\in\mathcal{A}} \mathbb{E}_{a\sim\pi}\left[N_T(a)\right]\Delta_a$$

Total Regret Decomposition

$$L_T = \sum_{a \in \mathcal{A}} \mathbb{E}_{a \sim \pi} \left[N_T(a) \right] \Delta_a$$

- Basic result widely used in regret analysis of bandit algorithms.
- Interpretation? Build an algorithm using this conclusion?

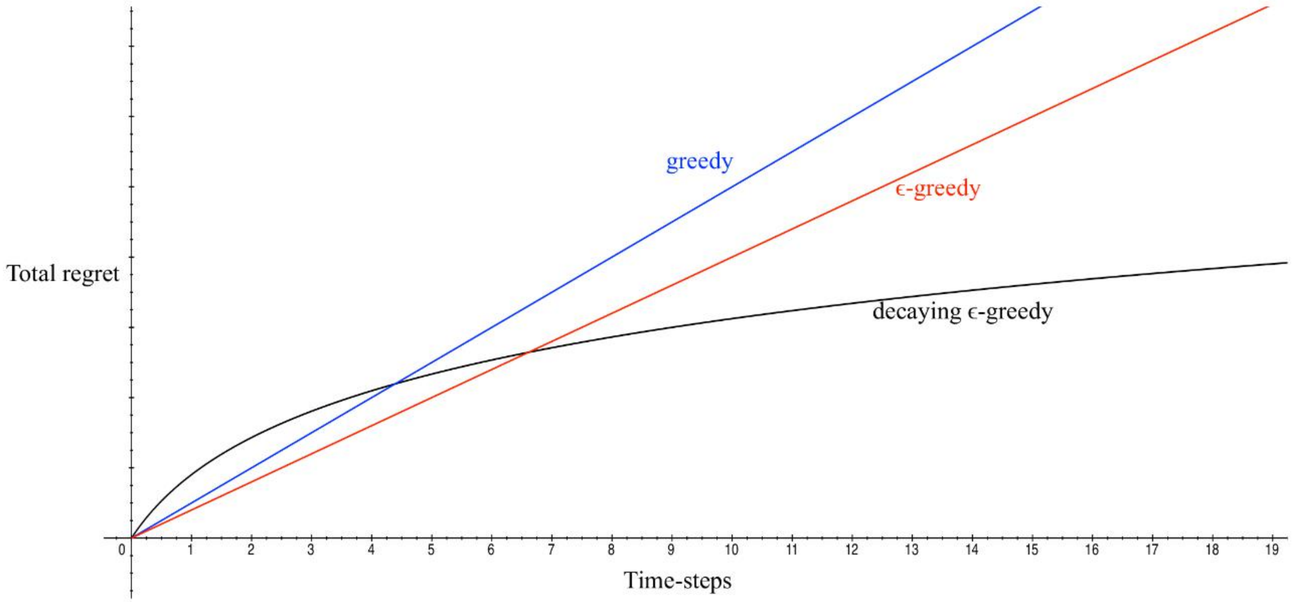
Total Regret Decomposition

$$L_T = \sum_{a \in \mathcal{A}} \mathbb{E}_{a \sim \pi} \left[N_T(a) \right] \Delta_a$$

- Basic result widely used in regret analysis of bandit algorithms.
- Interpretation? Build an algorithm using this conclusion?

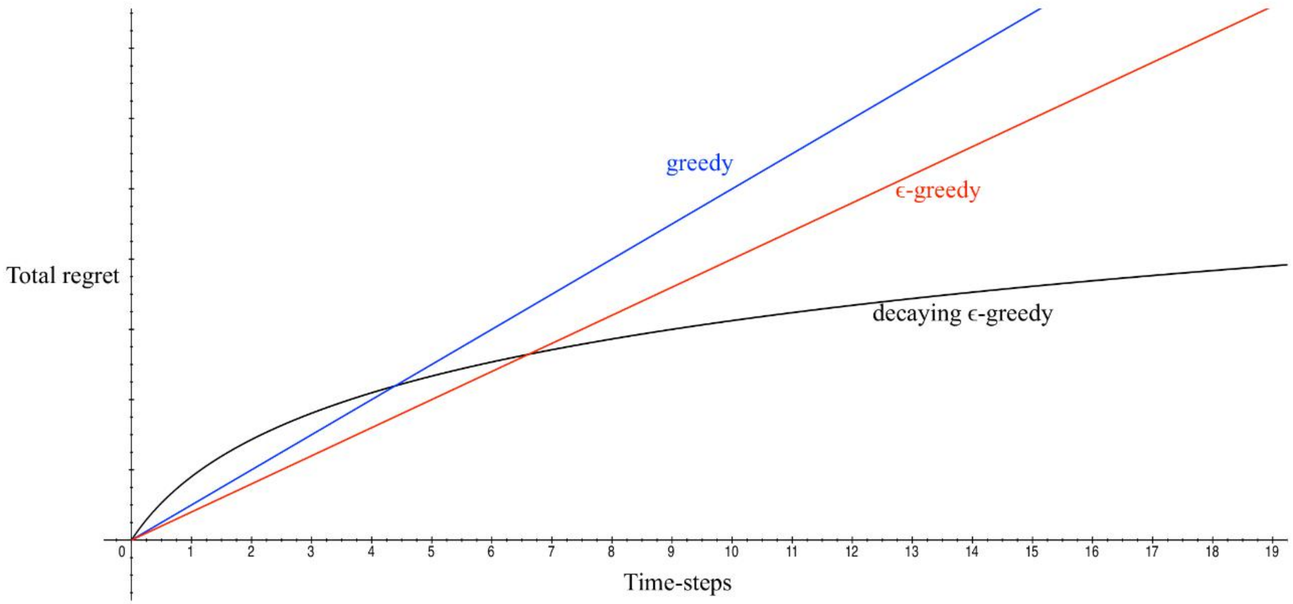
- Interpretation: a good algorithm picks actions with large gaps less frequently.
- Issue: The gap is not available, since it requires knowing the optimal value V^*.

Desirable Total Regret Behavior



- What can you infer from this figure?

Desirable Total Regret Behavior



- What can you infer from this figure?
- Ideally, for the total regret, over time:
  - the rate of increase quickly decreases
  - converge to a small value
- For example, decaying ϵ-greedy (to introduce soon) has logarithmic asymptotic total regret.

Greedy and ϵ-Greedy

- Estimate the value of each action by Monte-Carlo estimation:

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t'=1}^{t} r_{t'} 1_{A_{t'}=a}$$

- Greedy algorithm selects action with the highest estimated value:

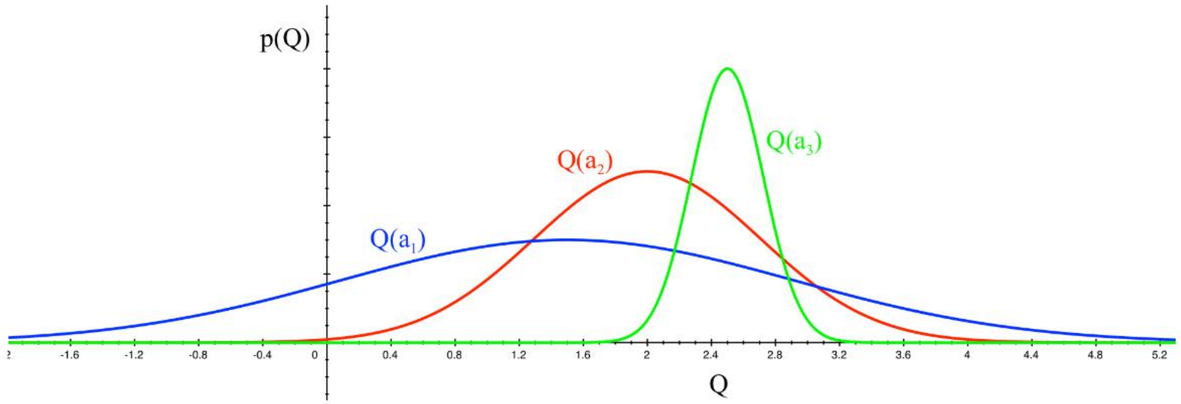
$$a_t = \operatorname*{argmax}_{a \in \mathcal{A}} \hat{Q}_t(a)$$

- ϵ-Greedy algorithm selects a random action with probability ϵ.
- Both algorithms have linear total regret because Greedy algorithm never explores and ϵ-Greedy algorithm forever explores.
- How to strike a balance? Can we achieve sublinear total regret?

Decaying ϵ-Greedy Algorithm

- By decaying parameter ϵ in ϵ-Greedy, it is possible to have logarithmic asymptotic total regret.
- Problem: the decaying schedule requires knowing the gap $\Delta_a = V^* - Q(a), \forall a \in \mathcal{A}$, which is not available.
- How to achieve logarithmic asymptotic total regret without using intractable terms?

The Principle of Optimism in the Face of Uncertainty



- Given 3 actions a_1, a_2, a_3 with the estimated values as shown.
- Which action should the policy pick to find the optimal action?
- The more uncertain the value estimate of an action is, the more important to take that action.
- In this case, actions a_1 and a_2 have:
  - more uncertain value estimates than a_3.
  - non-trivial probability of having higher value than a_3.

Upper Confidence Bound

- This principle is operationalized into upper confidence for each action value:
  - For each action a, estimate an upper confidence $\hat{U}_t(a)$.
  - Such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with high probability.
- $\hat{Q}_t(a) + \hat{U}_t(a)$ is called the Upper Confidence Bound (UCB) of action a.
- Select action maximizing the Upper Confidence Bound.

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, \hat{Q}_t(a) + \hat{U}_t(a)$$

Deriving UCB1 Algorithm

- Recall the (weak) law of large numbers: For any positive number ϵ

$$\lim_{n \to \infty} \Pr\Big(|\overline{X}_n - \mu| > \varepsilon \Big) = 0$$

- Interpretation: Empirical average approaches expectation for infinite samples.
- Given finite samples, how far would the empirical average derivate from the expectation?

Deriving UCB1 Algorithm

- Hoeffding's inequality: Let $X_1, \ldots, X_t$ be i.i.d. random variables in $[0, 1]$, and let $\bar{X}_t = \frac{1}{t} \sum_{t'=1}^{t} X_{t'}$ be the sample mean. Then:

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

- Applying the inequality to the value estimates of the actions:

$$\mathbb{P}\left[Q(a) > \hat{Q}_t(a) + U_t(a)\right] \leq e^{-2N_t(a)U_t(a)^2}, \ \forall a \in \mathcal{A}$$

- Assume we know r_{max}, r_{min} for each action and can scale the reward accordingly such that $r \in [0, 1]$.
- Interpretation: $\left[-\infty, \hat{Q}_t(a) + U_t(a)\right]$ is the interval that $Q(a)$ will fall in with high probability.

Deriving UCB1 Algorithm

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

- We are interested in finding $U(t)$ so that there is only a small chance that the interval does not cover the true $Q(a)$.
- If the upper bound of this chance is already below p, then the interval must be wide enough:

$$\begin{aligned} e^{-2N_t(a)U_t(a)^2} &\leq p \\ \implies U_t(a) &\geq \sqrt{\frac{-\log p}{2N_t(a)}} \end{aligned}$$

- Interpretation: $Q_t(a)$ falls in $\hat{Q}_t(a) + U_t(a)$ with probability larger than $1 - p$.
- If we would the interval to be tight, then we pick the smallest possible $U_t(a)$.
- Therefore, we take $\hat{Q}_t(a) + \sqrt{\frac{-\log p}{2N_t(a)}}$ as an optimistic estimation of $Q_t(a)$.

Deriving UCB1 Algorithm

- We take $\hat{Q}_t(a) + \sqrt{\frac{-\log p}{2N_t(a)}}$ as an *optimistic* estimation of $Q_t(a)$.
- To make the numbers concrete, let us assume a decaying p over time: $p = \frac{1}{t^4}$
- This leads to the UCB1 algorithm:

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, \hat{Q}_t(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

- Interpretation?

Deriving UCB1 Algorithm

- We take $\hat{Q}_t(a) + \sqrt{\frac{-\log p}{2N_t(a)}}$ as an *optimistic* estimation of $Q_t(a)$.
- To make the numbers concrete, let us assume a decaying p over time: $p = \frac{1}{t^4}$
- This leads to the UCB1 algorithm:

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, \hat{Q}_t(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

- Interpretation?
  - Interpretation: $N_t(a)$ serves as a proxy for how uncertain the algorithm
  - Large $N_t(a) \implies$ the policy has taken $N_t(a)$ many times $\implies$ Less uncertain value estimate.
  - Small $N_t(a) \implies$ the policy has not taken $N_t(a)$ many times $\implies$ More uncertain value estimate.

Theoretical Properties of UCB1 Algorithm

- For any timestep T, the total regret of UCB1 is at most *logarithmic* in the number of timestep T:

$$L_T \leq 8 \ln T \underbrace{\left[\sum_{a:\Delta_a>0} \left(\frac{1}{\Delta_a} \right) \right]}_{const} + \underbrace{\left(1 + \frac{\pi^2}{3} \right) \left(\sum_a \Delta_a \right)}_{const}$$

Finite-time Analysis of the Multiarmed Bandit Problem. Auer et al.

Limitation of Exact Count-based Methods

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \ \hat{Q}_t(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

- The upper confidence bound is an inverse function of the square root of the count $N_t(a)$.
- Possible to extend count-based methods to tabular MDP:
  - Either uses state visitation count $N_t(s)$ or state-action visitation count $N_t(s, a)$
  - The state visitation count is essential for many theoretical analysis of exploration
- However, it is only possible to compute the exact count for low-dim state space.
- For high-dim state space (such as images):
  - Most states will have count 0
  - Most states will be equally novel by exact count metric
  - But the agent cannot visit all states due to high-dim nature of state space
- We need an approximate count that generalizes across states:
  - A state should have high approximated count if it is similar to previously visited states

As an Aside: Density Estimation Problem

- Consider a random variable sampled from an arbitrary distribution:

$$\boldsymbol{X} \sim \mathbb{P}(\cdot)$$

- Given samples from the distribution:

$$\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} = \boldsymbol{x}_{1:n}$$

- Given an arbitrary sample $\boldsymbol{x}$, the *density estimation problem* queries the probability of $\boldsymbol{x}$:

$$\mathbb{P}(\boldsymbol{x}|\boldsymbol{x}_{1:n}) \approx \mathbb{P}(\boldsymbol{x})$$

Density Estimation over State Space

- Consider an MDP with a *countable* state space $\mathcal{S}$, denote a sequence of t states by $s_{1:t}$.
- Given $s_{1:t} \sim (\mathcal{M}, \pi)$, a density model over the state space $\mathcal{S}$ is:

$$\rho_t(s) = \rho(s; s_{1:t}) \approx \mathbb{P}(s|\mathcal{M}, \pi)$$

  - a good density model approximates $\mathbb{P}(s|\mathcal{M}, \pi)$ well
- For example,
  - the empirical density estimation is:

$$\rho_t(s) = \frac{\hat{N}_t(s)}{t}$$

  - we can also build (or learn) some density estimator that allows to predict the density for unseen states, e.g., convert to features and then do density estimation by Gaussian Mixture Model (GMM).

Computing Pseudo-Count from Density Model

We introduce a hack to estimate a "pseudo-count" at any state s using a density model $\rho_t(s)$ from visited states $s_{1:t}$:

- First, given $s_{1:t}$, we can estimate the density of some state s by

$$\rho_t(s) = \rho(s; s_{1:t})$$

- Next, we "imagine" that next step we will obtain one more sample of s, i.e., the visited state sequence becomes $\{s_{1:t}, s\}$. Then, the density of s will be

$$\rho_{t+1}(s) = \rho(s; s_{1:t}, s)$$

Computing Pseudo-Count from Density Model

- According to the empirical density estimation formula, we "expect" that the *pseudo count* of state visitation $\hat{N}_t$ to satisify

$$\rho_t(s) = \frac{\hat{N}_t(s)}{t}, \qquad \rho_{t+1}(s) = \frac{\hat{N}_t(s)+1}{t+1}$$

- Cancel t from the equations, and we can solve $\hat{N}_t(s)$:

$$\hat{N}_t(s) = \frac{\rho_t(s)\left(1-\rho_{t+1}(s)\right)}{\rho_{t+1}(s)-\rho_t(s)}$$

- The new MDP reward function is:

$$R(x,a) + \sqrt{\frac{\beta}{\hat{N}_t(s)+0.01}}$$

Unifying Count-Based Exploration and Intrinsic Motivation. Bellemare et al.

Pseudo-Count Performance

- In 100 million frames of training:
  - DQN explores 2 rooms
  - DQN + pseudo-count explores 15 different rooms
- Go to 1:50 mark in the video to see exploration behavior.

Intrinsic Rewards

The Perspective of Intrinsic Rewards

- UCB1 chooses actions by estimating an upper confidence for each action value:

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, \hat{Q}_t(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

- Since the squared-root term is added to $\hat{Q}$, it can be viewed as a reward!
- Pseudo-count methods also add a manually designed reward term to the MDP reward function

Adding another term to supplement the MDP reward is a common technique. The added term is sometimes referred as "intrinsic rewards".

Intrinsic Rewards

- In addition to the pseudo-count, there are many other ways to compute an intrinsic reward:
  - Random Network Distillation
  - Curiosity-driven exploration through:
    - forward dynamics prediction model
    - inverse dynamics model
  - Entropy regularization
  - etc ...

Novelty-driven Exploration without Estimating Pseudo-count

- If the MDP reward is often 0, add another term to encourage the policy to visit novel states.

$$r_t = e_t + i_t$$

- where:
  - e_t is the reward defined by the MDP, often called extrinsic reward
  - i_t is the novelty bonus, often called intrinsic reward
- We will use Random Network Distillation as a case study.

Random Network Distillation

- Use two neural networks to compute the novelty bonus (intrinsic reward i_t):
  - A fixed and randomly initialized target network, mapping observation to k-dim features, $f : \mathcal{O} \to \mathbb{R}^k$
  - A predictor network, $\hat{f}_\theta : \mathcal{O} \to \mathbb{R}^k$
- Given a state s_t, the novelty bonus is the difference in predicted features.

$$i_t = \|\hat{f}_\theta(s_t) - f(s_t)\|^2$$

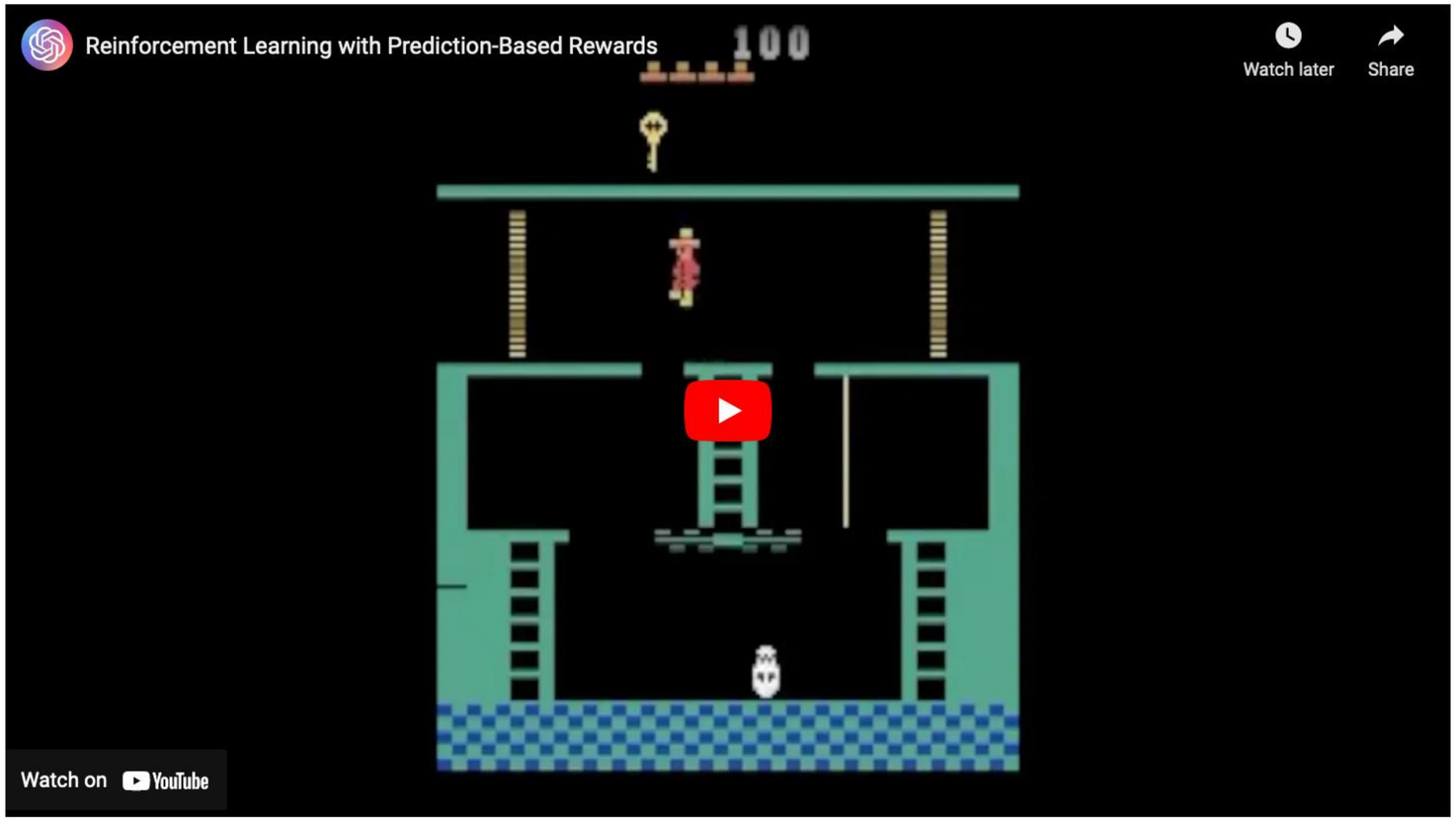
- The predictor network is trained to match the output of the target network using previously collected experience:

$$\text{minimize}_\theta \|\hat{f}_\theta(s) - f(s)\|^2$$

  Usually we only optimize for a few steps.

Random Network Distillation. Burda, Edwards, Storkey, Klimov.

Random Network Distillation Performance



End