Exploration in RL: Contextual & Bayesian Bandits, Thompson Sampling

Roberto Capobianco

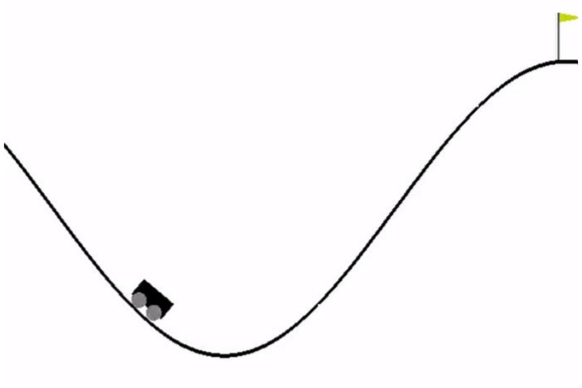
Sapienza Università di Roma

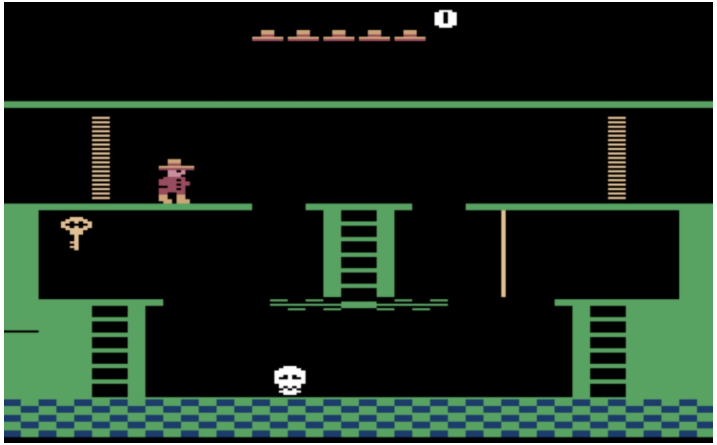
Adapted from Wen Sun's slides

Recap

Failure Mode of RL

Sparse rewards (e.g., Mountain Car or Montezuma’s Revenge) are a problem in RL: zero reward everywhere except in few states





Exploration: the Big Pain of RL

We need to carefully and systematically explore (remember states we visited, and try to visit unexplored regions)

Exploration-Exploitation Trade-off: should we make the best decision given current information, or should we collect more information? In other words: should I sacrifice something now to get more in the future? (chicken-egg problem)

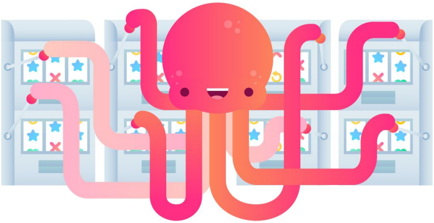
e.g., go to my favourite restaurant vs try a new one

Multi-Armed Bandit

Let’s consider a simplified MDP to analyze exploration: **Multi-Armed Bandits**

- One single state
- K different arms (think of them as actions): a_1, ..., a_k
- Each arm has unknown reward distribution ν_i with mean $\mu_i = \mathbb{E}_{r\sim\nu_i}[r]$
- Every time we pull an arm we observe an i.i.d. reward

Multi-Armed Bandit: Interaction



The interactive process that we deal with in MAB is the following:

For $t = 0, \ldots, T-1$:

1. Pull an arm I_t in $\{1, \ldots, K\}$ based on historical information
2. Observe i.i.d. reward $r_i \sim \nu_i$ of arm I_t (we do not observe rewards of untried arms)

But what are we trying to optimize exactly? **REGRET!**

Regret

We want to minimize our **opportunity loss**, which is expressed in the form of the regret

The regret is the **total expected reward if we pull the best arm for T rounds** VS the **total expected reward of the arms we pulled over T rounds**

$$\mathrm{Regret}_T = \boxed{T\mu^\star} - \boxed{\sum_{t=0}^{T-1} \mu_{I_t}} \qquad \mu^\star = \max_{i\in[K]} \mu_i$$

Greedy Algorithm

Algorithm:

- try each arm once
- commit to the one that has the highest observed reward

Problem: a (bad) arm with low μ_i may generate a high reward by chance, as we sample $r_i \sim \nu_i$ and it's i.i.d.

Consider two arms a_1, a_2: Reward dist for a_1: prob 60%: 1, else 0; for a_2: prob 40% 1, else 0. Now: a_1 is clearly better but with prob 16% we can observe (0, 1)

Explore & Commit Algorithm

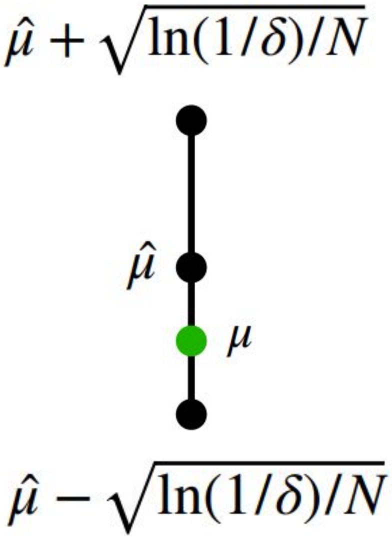
1. Set N = T/K, where T >> K and K is the number of arms
2. For k = 1, ..., K: **(explore)**
   - pull arm *k* for N times
   - observe the set $\{r_i\}_{i=1}^{N} \sim \nu_i$
   - compute the empirical mean $\hat{\mu}_k = \sum_{i=1}^{N} r_i / N$
3. For t = NK, ..., T: **(commit)**
   - pull the best empirical arm

$$I_t = \arg\max_{i \in [K]} \hat{\mu}_i$$

Hoeffding Inequality

Do we have a confidence interval on our empirical mean? <u>During exploration, for each arm</u>, given a distribution with mean μ and N i.i.d. samples, we have with probability 1-δ:

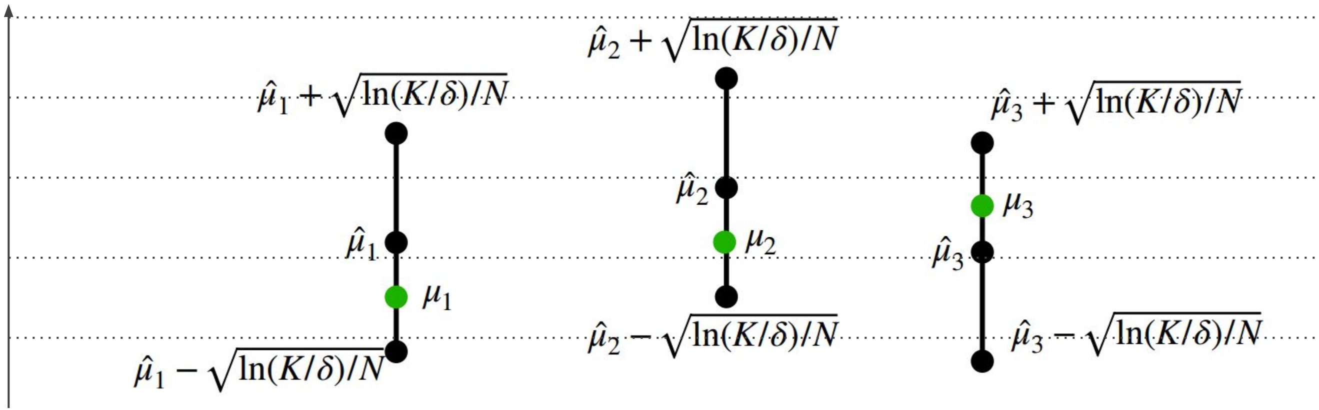
$$\left|\sum_{i=1}^{N} r_i/N - \mu_i\right| \leq O\left(\sqrt{\frac{\ln(1/\delta)}{N}}\right)$$



e.g., δ = 0.01, confidence bound holds with probability 99%

Hoeffding Inequality

Do we have a confidence interval on our empirical mean? During exploitation, for all arms, given a distribution with mean μ and N i.i.d. samples, we have with probability 1-δ:



Regret Calculation

- Empirical best arm:

$$\hat{I} = \arg\max_{i \in [K]} \hat{\mu}_i$$

- Best arm:

$$I^\star = \arg\max_{i \in [K]} \mu_i$$

Total regret: $\text{Regret}_T = \text{Regret}_{explore} + \text{Regret}_{exploit} \leq NK + 2T\sqrt{\dfrac{\ln(K/\delta)}{N}}$

To minimize our regret, we want to optimize N: take the gradient of the regret, set it to 0, solve for N

Regret Calculation

- Empirical best arm:

$$\hat{I} = \arg\max_{i \in [K]} \hat{\mu}_i$$

- Best arm:

$$I^{\star} = \arg\max_{i \in [K]} \mu_i$$

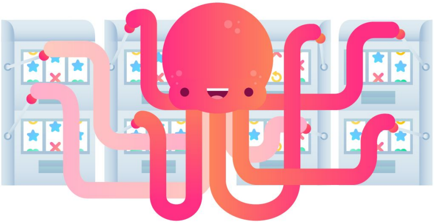
Total regret: $\text{Regret}_T = \text{Regret}_{explore} + \text{Regret}_{exploit} \leq NK + 2T\sqrt{\frac{\ln(K/\delta)}{N}}$

$$N = \left(\frac{T\sqrt{\ln(K/\delta)}}{2K} \right)^{2/3} \qquad \text{Regret}_T \leq O\left(T^{2/3} K^{1/3} \cdot \ln^{1/3}(K/\delta)\right)$$

Approaches 0 as T goes to infinite

SAPIENZA
UNIVERSITÀ DI ROMA

Regret Decaying

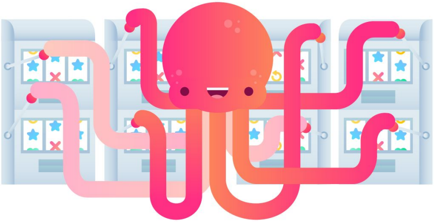


The decaying rate of the regret using the explore & commit algorithm is kind of slow ($T^{2/3}$). Can we get something faster, like $O(\sqrt{T})$?

$O(\sqrt{T})$ is actually the minimum we can get as it is a lower bound (no algorithm ever will be faster than this)

Let’s try to design a new algorithm

Statistics to Maintain & Confidence



Let’s write a list of generic statistics that we need to maintain in order to compute our confidence bounds and the regret

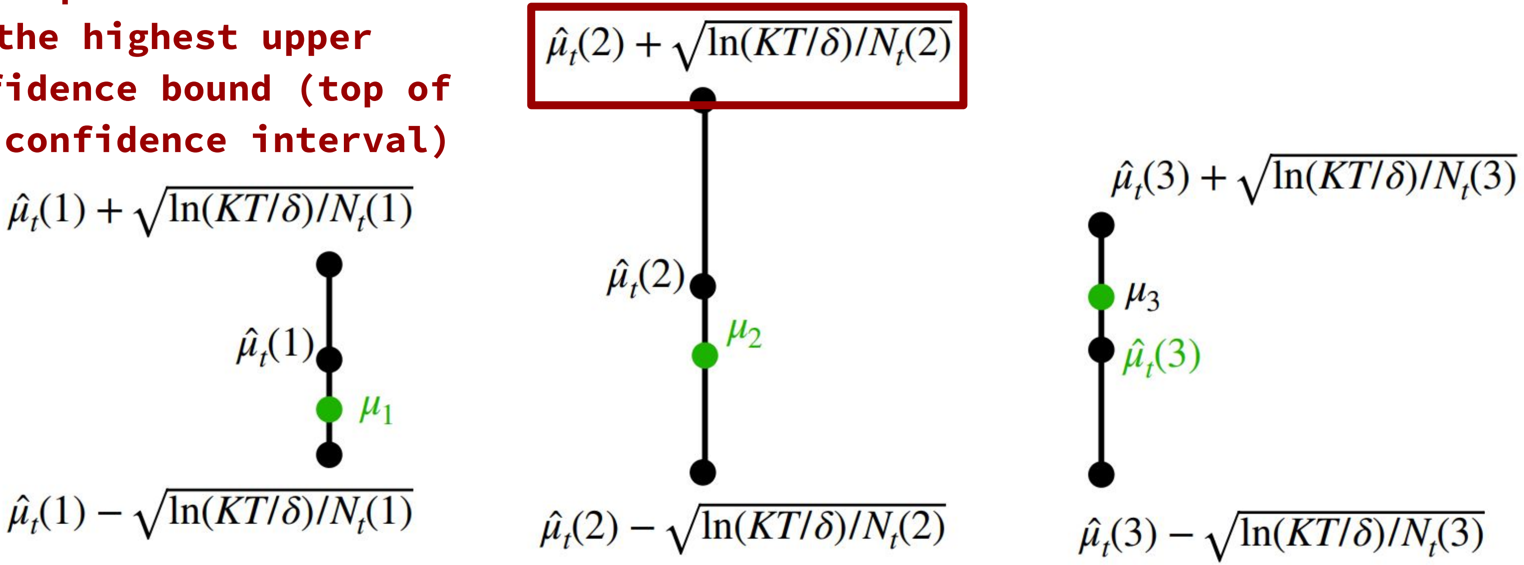
- # of times we have tried arm i $N_t(i) = \sum_{\tau=0}^{t-1} \mathbf{1}\{I_\tau = i\}$

- empirical mean so far $\hat{\mu}_t(i) = \sum_{\tau=0}^{t-1} \mathbf{1}\{I_\tau = i\} r_\tau / N_t(i)$

Confidence with probability 1-δ: $|\hat{\mu}_t(i) - \mu_i| \leq \sqrt{\frac{\ln(KT/\delta)}{N_t(i)}}$

Optimism in the Face of Uncertainty

Let's pick the arm with the highest upper confidence bound (top of the confidence interval)



UCB Algorithm

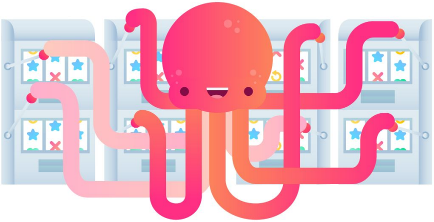
- For the first K iterations, pull each arm once
- For t = K, ..., T:
  - pick the action with the highest upper confidence bound

$$I_t = \arg\max_{i \in [K]} \left(\hat{\mu}_t(i) + \boxed{\sqrt{\frac{\ln(KT/\delta)}{N_t(i)}}} \right)$$

  - update statistics

Reward bonus is high if we did not try action many times: exploration

UCB Algorithm: Regret



$$\text{Regret-at-t} = \mu^\star - \mu_{I_t} \leq \hat{\mu}_t(I_t) + \sqrt{\frac{\ln(TK/\delta)}{N_t(I_t)}} - \mu_{I_t} \leq 2\sqrt{\frac{\ln(TK/\delta)}{N_t(I_t)}}$$

$$\text{Regret}_T = \sum_{t=0}^{T-1}\left(\mu^\star - \mu_{I_t}\right) \leq \sum_{t=0}^{T-1} 2\sqrt{\frac{\ln(TK/\delta)}{N_t(I_t)}} \leq 2\sqrt{\ln(TK/\delta)} \cdot \sum_{t=0}^{T-1}\sqrt{\frac{1}{N_t(I_t)}}$$

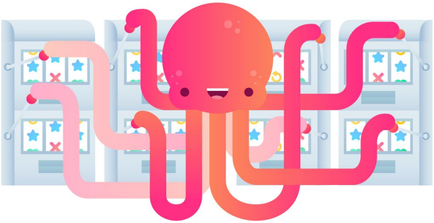
$$\sum_{t=0}^{T-1}\sqrt{\frac{1}{N_t(I_t)}} \leq O\left(\sqrt{KT}\right) \longrightarrow$$ **With high probability** $$\text{Regret}_T = \widetilde{O}\left(\sqrt{KT}\right)$$

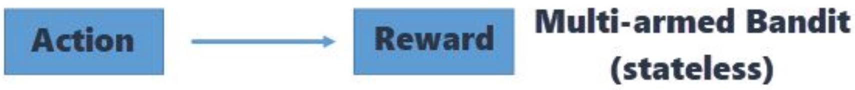
See also
https://wensun.github.io/CS4789_data/UCB_note_new.pdf

SAPIENZA
UNIVERSITÀ DI ROMA

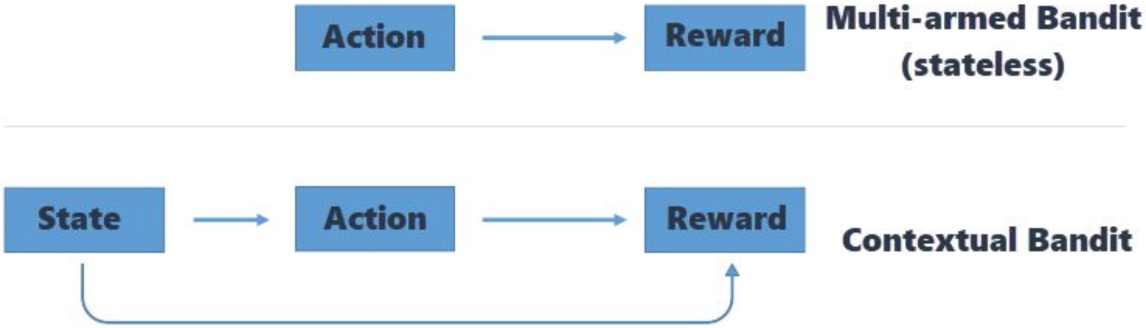
End Recap

From Multi-Armed to Contextual Bandits





From Multi-Armed to Contextual Bandits



Contextual bandits add back some context (state)

Contextual Bandits: Interaction

The interactive process that we deal with in CB is the following:

For $t = 0, \ldots, T-1$:

1. A new i.i.d. context x_t in X appears
2. Select an action a_t in A based on historical information and context
3. Observe reward $r(x_t, a_t)$ (which is context and arm dependent)

Contextual Bandits: Interaction

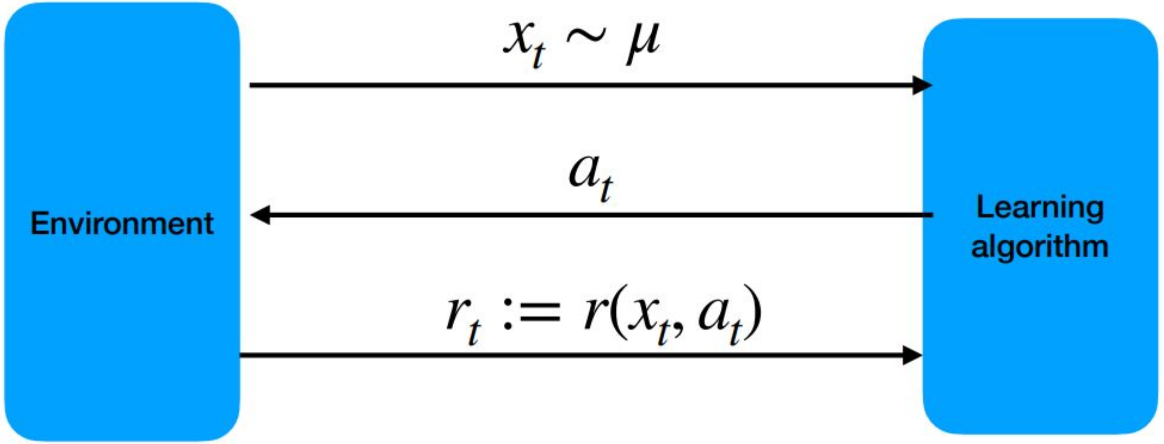
The interactive process that we deal with in CB is the following:

For $t = 0, \ldots, T-1$:

1. A new i.i.d. context x_t in X appears
2. Select an action a_t in A based on historical information and context
3. Observe reward $r(x_t, a_t)$ (which is context and arm dependent)

For simplicity we assume deterministic rewards, as the context is the challenge here

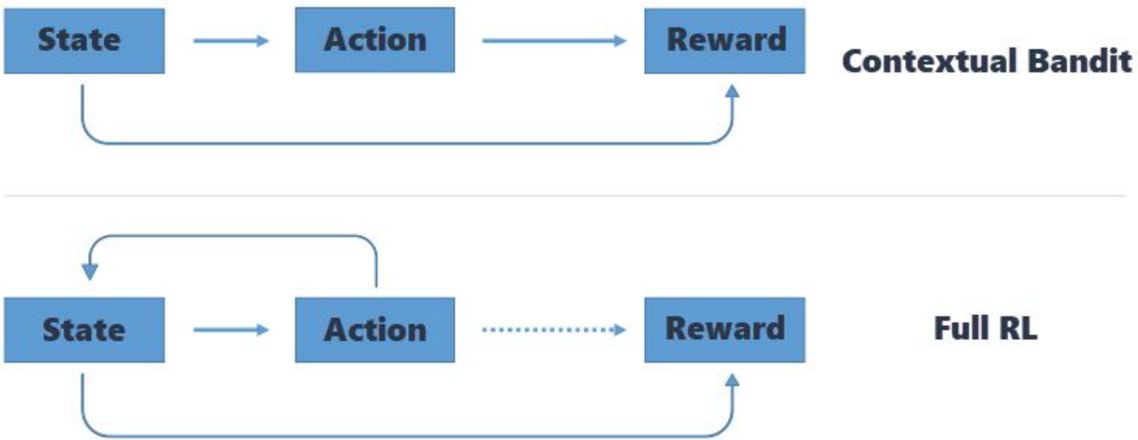
Contextual Bandits: Interaction



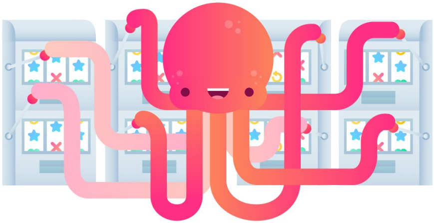
After we get reward, we start a new iteration: states do not depend on previous actions, they are just sampled in i.i.d. fashion

Contextual Bandits VS RL

In RL, conversely, states depend on previous actions: **we can say that contextual bandits are Finite-Horizon MDPs with horizon 1**



Contextual Bandits: Example



One domain of application of contextual bandits is recommendation systems:

- Context corresponds to user information (e.g., age, height, weight, job, etc.)
- Arms correspond to items to recommend (e.g., news, movies, etc.)
- Each arm has a click-through-rate (0/1 reward based on click) that we aim to maximize

How do we decide which item to propose next, **in personalized way**?

Contextual Bandits: Regret

Optimal policy: $\pi^\star = \arg\max_{\pi \in \Pi} \mathbb{E}_{x \sim \mu} r(x, \pi(x))$

At every iteration $a_t = \pi_t(x_t)$ is selected and a reward $r(x_t, a_t)$ is received: the regret is the **total expected reward if we always use π^*** VS the **total expected reward if we use our learned sequence of policies**

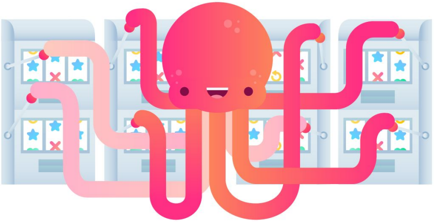
$$\text{Regret}_T = \boxed{T\mathbb{E}_{x\sim\mu}[r(x, \pi^\star(x))]} - \boxed{\sum_{t=0}^{T-1} \mathbb{E}_{x\sim\mu}[r(x, \pi^t(x))]}$$

Note that policies are different at every iteration t

Explore & Commit Algorithm

1. For t = 0, ..., N-1: **(explore)**
   - observe state $x_t \sim \mu$
   - uniform-randomly sample $a_t \sim \mathrm{Unif}(A)$
   - observe reward $r_t = r(x_t, a_t)$
   - build, for x_t, an unbiased estimate of $\mathbb{E}_{a_t \sim p} \hat{\mathbf{r}}[a] = r(x_t, a), \forall a$
2. Compute policy
$$\hat{\pi} = \arg\max_{\pi \in \Pi} \sum_{i=0}^{N-1} \hat{\mathbf{r}}_i[\pi(x_i)]$$
3. For t = N, ..., T-1: **(commit)**
   - observe state $x_t \sim \mu$
   - play arm $a_t = \hat{\pi}(x_t)$

Explore & Commit Algorithm



1. For t = 0, ..., N-1: **(explore)**
   - observe state $x_t \sim \mu$
   - uniform-randomly sample $a_t \sim \mathrm{Unif}(A)$
   - observe reward $r_t = r(x_t, a_t)$
   - build, for x_t, an unbiased estimate of $\boxed{\mathbb{E}_{a_t \sim p} \hat{\mathbf{r}}[a] = r(x_t, a), \forall a}$
2. Compute policy

$$\hat{\pi} = \arg\max_{\pi \in \Pi} \sum_{i=0}^{N-1} \hat{\mathbf{r}}_i[\pi(x_i)]$$

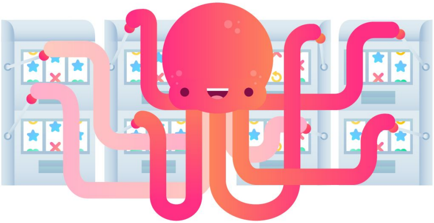
If we know $p(a_t)$, given we sample from p

3. For t = N, ..., T-1: **(commit)**
   - observe state $x_t \sim \mu$
   - play arm $a_t = \hat{\pi}(x_t)$

$$\hat{\mathbf{r}}[a] = \frac{r(x_t, a)\mathbf{1}[a = a_t]}{p(a_t)} \qquad \begin{bmatrix} 0 \\ 0 \\ \cdots \\ r_t / p(a_t) \\ 0, \\ \cdots \\ 0 \end{bmatrix}$$

SAPIENZA UNIVERSITÀ DI ROMA

Explore & Commit Algorithm



1. For t = 0, ..., N-1: **(explore)**
    - observe state $x_t \sim \mu$
    - uniform-randomly sample $a_t \sim \mathrm{Unif}(A)$
    - observe reward $r_t = r(x_t, a_t)$
    - build, for x_t, an unbiased estimate of $\boxed{\mathbb{E}_{a_t \sim p} \hat{\mathbf{r}}[a] = r(x_t, a), \forall a}$
2. Compute policy

$$\hat{\pi} = \arg\max_{\pi \in \Pi} \sum_{i=0}^{N-1} \hat{\mathbf{r}}_i[\pi(x_i)]$$

3. For t = N, ..., T-1: **(commit)**
    - observe state $x_t \sim \mu$
    - play arm $a_t = \hat{\pi}(x_t)$

Given we are sampling from Unif(A)

$$\hat{\mathbf{r}}_t[a] = \begin{cases} 0 & a \neq a_t \\ \frac{r_t}{1/|\mathscr{A}|} & a = a_t \end{cases}$$

Explore & Commit Algorithm: Regret

$$\text{Regret}_T = T\mathbb{E}_{x\sim\mu}[r(x,\pi^\star(x))] - \sum_{t=0}^{T-1} \mathbb{E}_{x\sim\mu}[r(x,\pi^t(x))] = O\left(T^{2/3}K^{1/3}\cdot \ln(|\Pi|)^{1/3}\right)$$

Regret also depends on the size of the space/class of policies that we are considering

ε-Greedy

Instead of setting a threshold for exploring and then committing, we can try to interleave exploration and exploitation

1. For t = 0, ..., T: **(interleave exploration & exploitation)**
   - observe state $x_t \sim \mu$
   - $a_t \sim p_t = \boxed{(1-\varepsilon)\delta(\pi^t(x_t))} + \boxed{\varepsilon \mathrm{Unif}(A)}$
   - observe reward $r_t = r(x_t, a_t)$
   - build, for x_t, an unbiased estimate of $\mathbb{E}_{a_t \sim p}\hat{\mathbf{r}}[a] = r(x_t, a), \forall a$
2. Update policy

$$\pi^{t+1} = \arg\max_{\pi \in \Pi} \sum_{i=0}^{t} \hat{\mathbf{r}}_i[\pi(x_i)]$$

ε = 0 -> exploit

ε = 1 -> uniformly explore

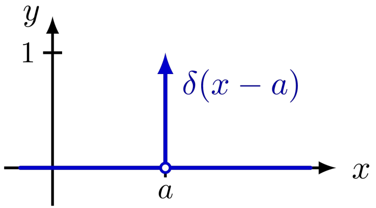
ε-Greedy

Instead of setting a threshold for exploring and then committing, we can try to interleave exploration and exploitation

1. For t = 0, ..., T: **(interleave exploration & exploitation)**
   - observe state $x_t \sim \mu$
   - $a_t \sim p_t = (1-\varepsilon)\boxed{\delta(\pi^t(x_t))} + \varepsilon \text{Unif}(A)$
   - observe reward $r_t = r(x_t, a_t)$
   - build, for x_t, an unbiased estimate of $\mathbb{E}_{a_t \sim p}\hat{\mathbf{r}}[a] = r(x_t, a), \forall a$
2. Update policy

$$\pi^{t+1} = \arg\max_{\pi \in \Pi} \sum_{i=0}^{t} \hat{\mathbf{r}}_i[\pi(x_i)]$$

Dirac delta function



ε-Greedy

Instead of setting a threshold for exploring and then committing, we can try to interleave exploration and exploitation

1. For t = 0, ..., T: **(interleave exploration & exploitation)**
    - observe state $x_t \sim \mu$
    - $a_t \sim p_t = (1-\varepsilon)\boxed{\delta(\pi^t(x_t))} + \varepsilon \mathrm{Unif}(A)$
    - observe reward $r_t = r(x_t, a_t)$
    - build, for x_t, an unbiased estimate of $\mathbb{E}_{a_t \sim p} \hat{\mathbf{r}}[a] = r(x_t, a), \forall a$
2. Update policy

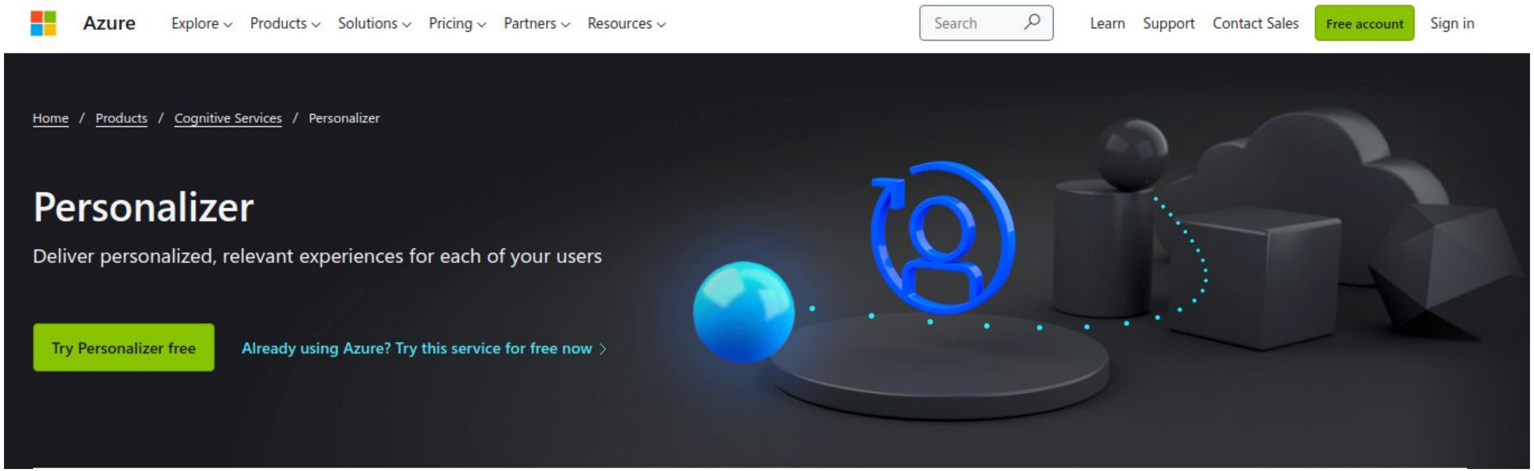
$$\pi^{t+1} = \arg\max_{\pi \in \Pi} \sum_{i=0}^{t} \hat{\mathbf{r}}_i[\pi(x_i)]$$

Step 3: Gradually decay ε

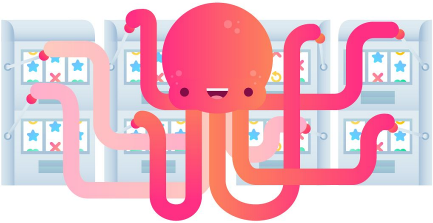
SAPIENZA UNIVERSITÀ DI ROMA

Real World Example

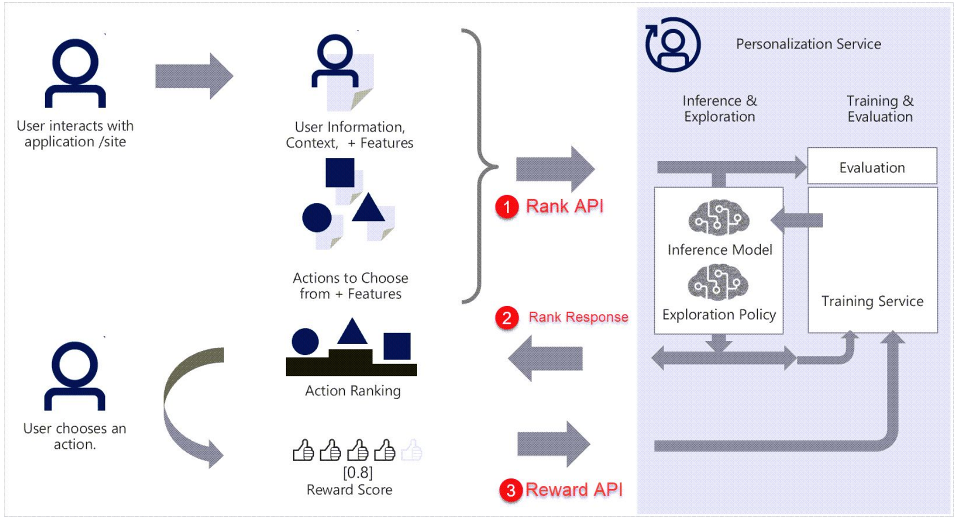
Algorithms for CB are actually used here:
https://azure.microsoft.com/en-us/products/cognitive-services/personalizer/



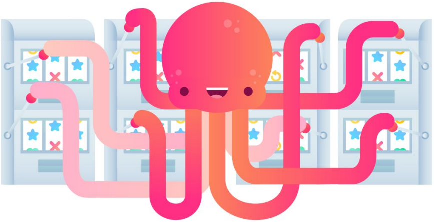
Real World Example



See https://learn.microsoft.com/en-us/azure/cognitive-services/personalizer/how-personalizer-works



Bayes Theorem Recall



This is the prior: i.e. what you believed before you saw the evidence.

This is the likelihood of seeing that evidence if your hypothesis is correct.

This is the posterior

$$p(H \mid D) = \frac{p(H)\,p(D \mid H)}{p(D)}$$

This is the normalizing constant:
i.e. The likelihood of that evidence under any circumstances.

Bayesian Bandits

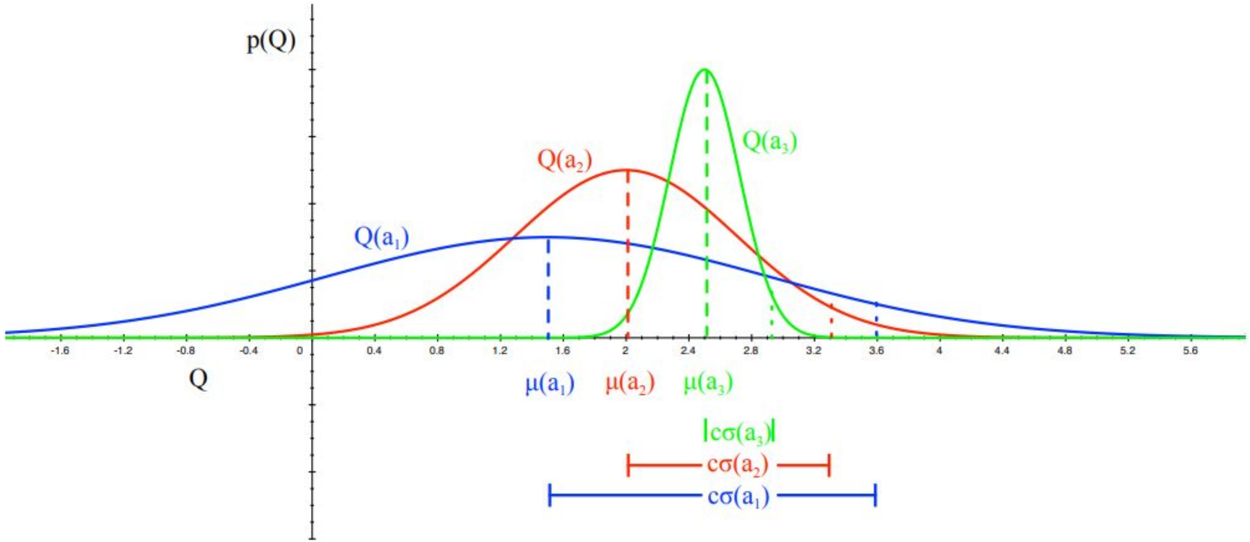
So far we have made no assumptions about the reward distribution ν_i, we only derived bounds on rewards

In Bayesian Bandits, however:

- We exploit *prior* knowledge of rewards
- Update a *posterior distribution* of rewards based on historical information
- Use posterior to guide exploration using:
  - upper confidence bounds (Bayesian UCB)
  - probability matching (Thompson Sampling)

Gaussian Bayesian Bandits Example

Assume ν_i is a Gaussian $N(\mu(i), \sigma^2(i))$



SAPIENZA
UNIVERSITÀ DI ROMA

Gaussian Bayesian Bandits Example

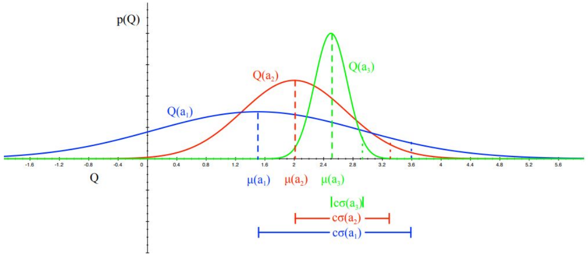
- Prior is often $N(0,\ 1)$
- Posterior update, given history h_t, is done as follows:

Product over all timesteps that select that action with corresponding parameters

$$p(\mu_t(i),\sigma^2_t(i)\,|\,h_t) \propto p(\mu_i,\sigma^2_i)\prod_{t|k=i}\boxed{N(r_t\,|\,\mu_{t-1}(k),\ \sigma^2_{t-1}(k))}$$

or more simply

$$p(\mu_t(i),\sigma^2_t(i)\,|\,r_t) \propto p(\mu(i),\sigma^2(i))N(r_t\,|\,\mu_{t-1}(i),\ \sigma^2_{t-1}(i))$$



SAPIENZA
UNIVERSITÀ DI ROMA

Gaussian Bayesian Bandits: UCB

Now we are modelling a distribution, so we already have confidence

What is confidence for Gaussians?

Gaussian Bayesian Bandits: UCB

Now we are modelling a distribution, so we already have confidence

What is confidence for Gaussians? **standard deviation**

Let’s do UCB by selecting the action with highest standard deviation

$$a_t = \text{argmax}_{i \text{ in } K}\, \mu_t(i) + c\sigma_t(i)/\sqrt{N_t(i)}$$

Gaussian Bayesian Bandits: Thompson Sampling

Thompson sampling is a way to do distribution matching: select action according to the probability that that action is optimal

- Optimistic in the face of uncertainty as uncertain actions have higher probability of satisfying maximization
- Uses sampling to avoid actual probability matching complications

Gaussian Bayesian Bandits: Thompson Sampling

For t = 0, ..., T:

1. for each arm i = 1, ..., K:
    - sample $\hat{\mathbf{r}}_i$ independently from $N(\mu_{t-1}(i), \sigma^2_{t-1}(i))$
2. pull arm

$$I_t = \arg\max_{i \in [K]} \hat{\mathbf{r}}_i$$

3. observe reward r_t
4. update posterior distribution $p(\mu_t(i), \sigma^2_t(i) | r_t)$

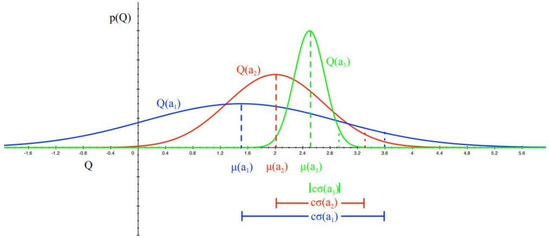
Gaussian Bayesian Bandits: Thompson Sampling

This is an estimation of the reward, in more generic MDPs this can be replaced with the Q function: we estimate a distribution of Q

For t = 0, ..., T:

1. for each arm i = 1, ..., K:
   - sample $\boxed{\hat{\mathbf{r}}_i}$ independently from $N(\mu_{t-1}(i), \sigma^2_{t-1}(i))$
2. pull arm

$$I_t = \arg\max_{i \in [K]} \hat{\mathbf{r}}_i$$



3. observe reward r_t
4. update posterior distribution $p(\mu_t(i), \sigma^2_t(i) | r_t)$

This can be done with different distributions as well

SAPIENZA
UNIVERSITÀ DI ROMA

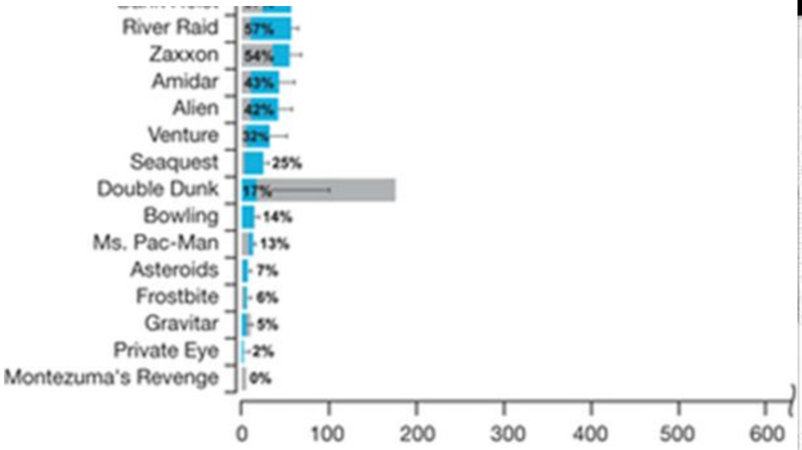
Thompson Sampling for Model-Based RL

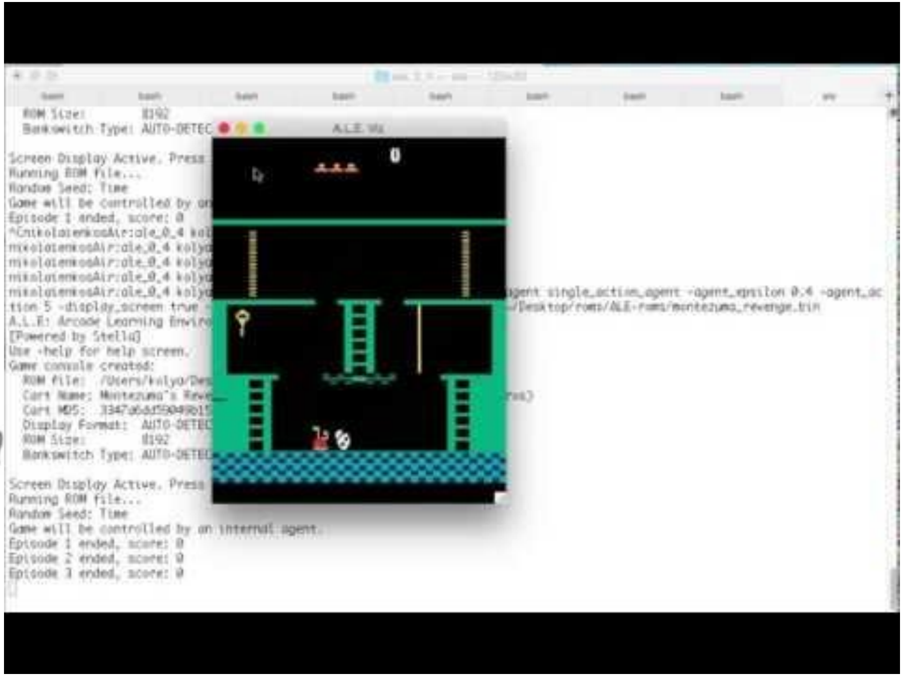
Initialize prior over the dynamics and reward models for each (s, a), $p(\mathcal{R}_{as})$, $p(\mathcal{T}(s'|s, a))$
Initialize state s_0
loop
    Sample a MDP $\mathcal{M}$: for each (s, a) pair, sample a dynamics model $\mathcal{T}(s'|s, a)$ and reward model $\mathcal{R}(s, a)$
    Compute $Q^*_{\mathcal{M}}$, optimal value for MDP $\mathcal{M}$
    $a_t = \arg\max_{a \in \mathcal{A}} Q^*_{\mathcal{M}}(s_t, a)$
    Observe reward r_t and next state s_{t+1}
    Update posterior $p(\mathcal{R}_{a_t s_t}|r_t)$, $p(\mathcal{T}(s'|s_t, a_t)|s_{t+1})$ using Bayes rule
    $t = t + 1$
end loop

Credits: Emma Brunskill

Exploration in State-of-the-art RL

Exploration Issues





Entropy

- Extra entropy term $H(\pi(a|s))$ in loss improve exploration
- Allows the policy to be ‘less committed’
  - Encourages diversity

Count-Based Exploration

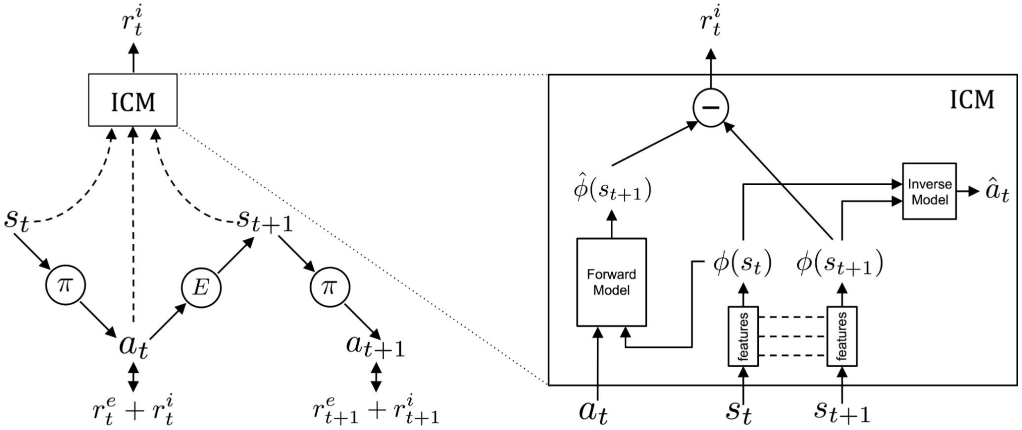
- Count number of times each state is visited
- Add extra term to the (extrinsic) reward
  - This term is intrinsic, so the full reward will be $r=r_i+r_e$
- Intrinsic reward term is higher for low-count states
- What if number of states is too large?
  - We need to approximate this count!

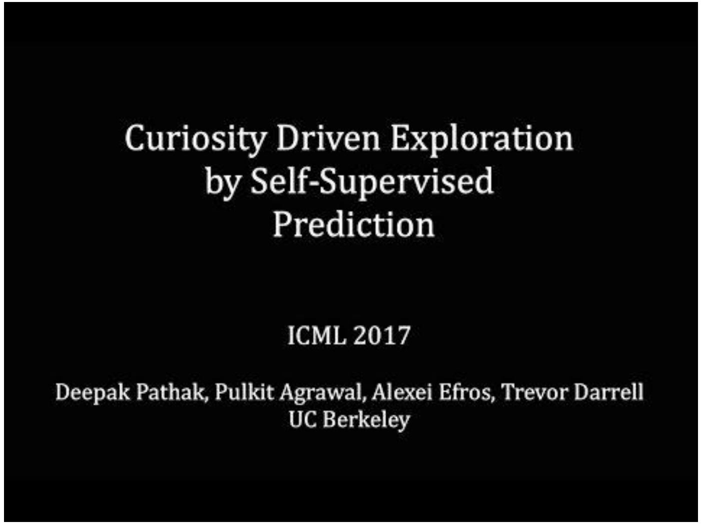
Count-Based Exploration

Bellemare, Marc G., et al. "Unifying count-based exploration and intrinsic motivation." *arXiv preprint arXiv:1606.01868* (2016).
One implementation here: https://github.com/RLAgent/state-marginal-matching

Curiosity-Driven Exploration

Pathak, Deepak, et al. "Curiosity-driven exploration by self-supervised prediction." *International Conference on Machine Learning*. PMLR, 2017.
Implementation: https://github.com/pathak22/noreward-rl

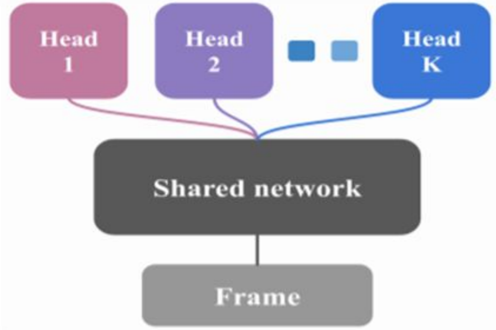




Thompson Sampling

Osband, Ian, et al. "Deep exploration via bootstrapped DQN." *arXiv preprint arXiv:1602.04621* (2016).

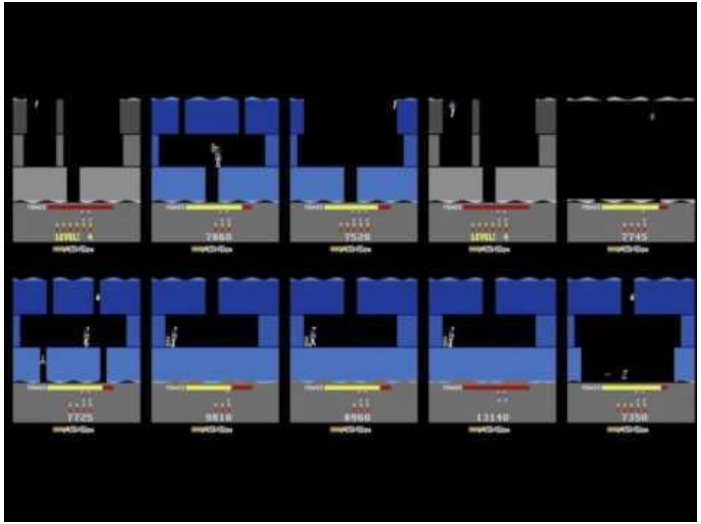
- **Thompson Sampling:** choosing the action that maximizes the expected reward with respect to a randomly drawn belief
- **Bootstrapping:** train K different heads (networks), each on a different subset of data



Bootstrapped DQN

Algorithm 1 Bootstrapped DQN

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



Exploration by Random Network Distillation

Burda, Yuri, et al. "Exploration by random network distillation." arXiv preprint arXiv:1810.12894 (2018)
Code: https://github.com/openai/random-network-distillation

- A target network, f, with fixed, randomized weights, which is never trained. That generates a feature representation for every state.

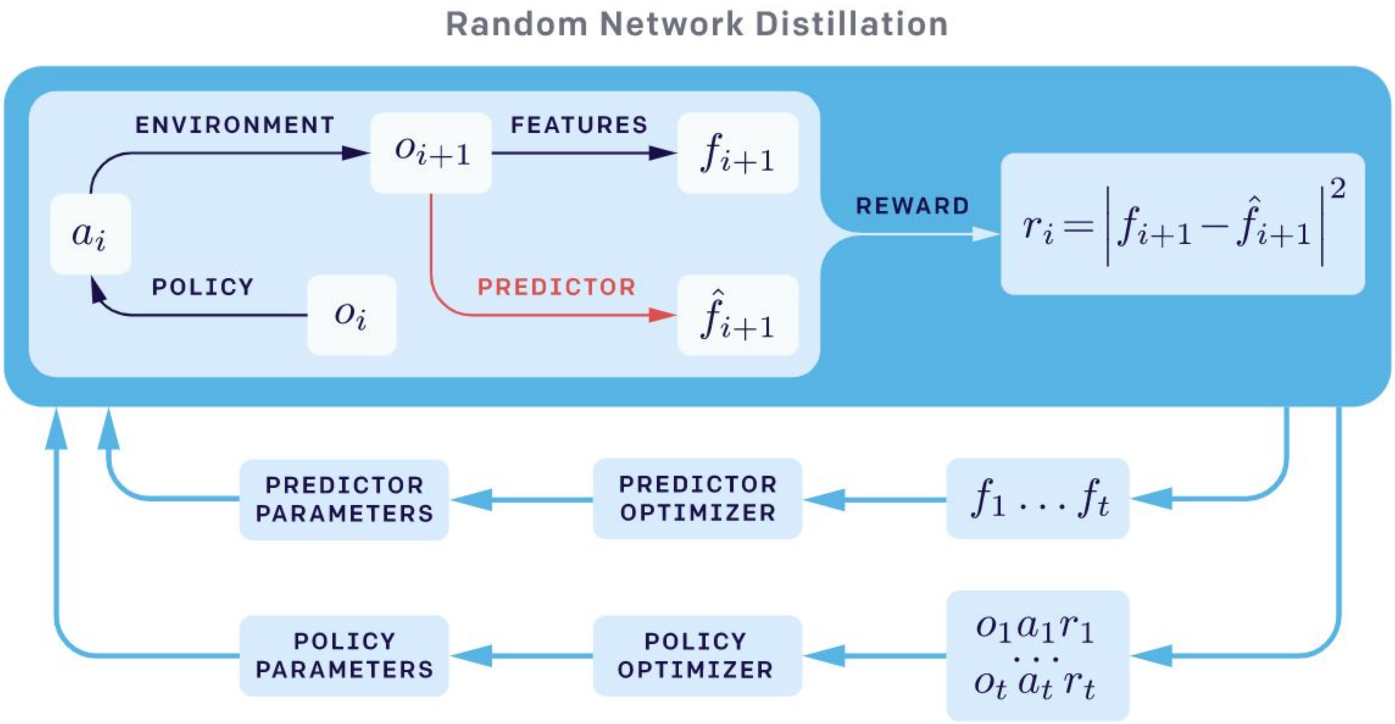
$$r_t^i = \underbrace{\left\| \hat{f}\left(s_{t+1}\right)}_{\text{Prediction of the Feature Representation of next state}} - \underbrace{f\left(s_{t+1}\right)}_{\text{Target Feature Representation of next state}} \right\|_{\boxed{2}}^{\boxed{2}}$$

L2 norm (to output a scalar)

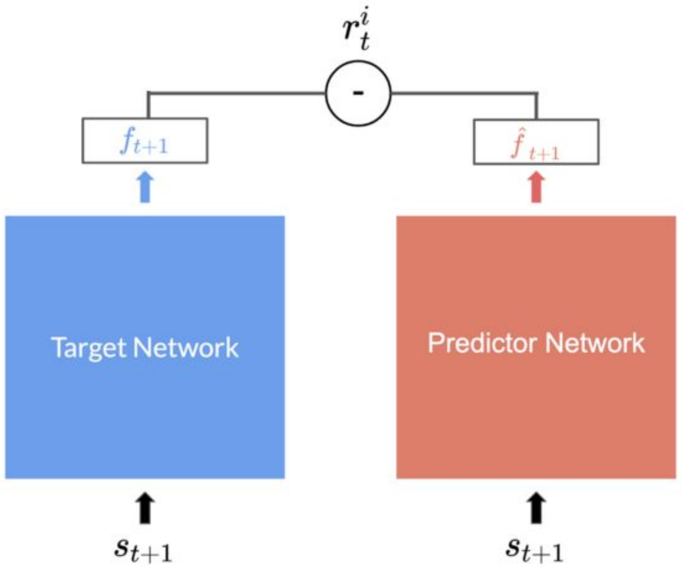
- A prediction network, f_hat, that tries to predict the target network's output

SAPIENZA
UNIVERSITÀ DI ROMA

Random Network Distillation



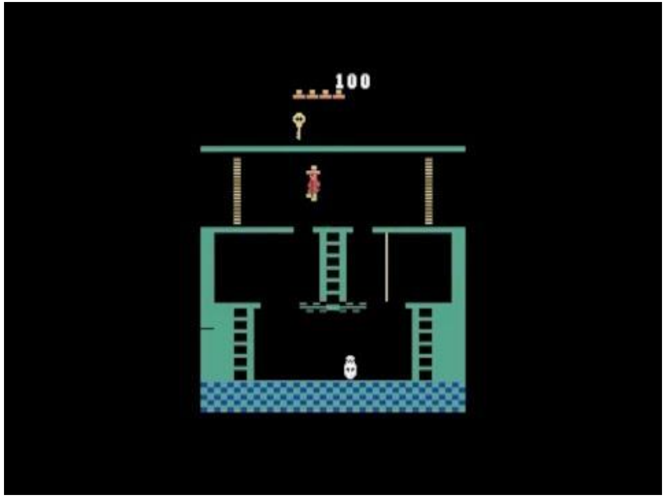
Random Network Distillation



Target network is randomly initialized

Target network will output a fixed feature representation of s_{t+1}

Predictor Network will tries to predict the target network's output $\hat{f}_{t+1}$



Never Give Up

Badia, Adrià Puigdomènech, et al. "Never give up: Learning directed exploration strategies." *arXiv preprint arXiv:2002.06038* (2020).

- At every step the current state embedding is added into M
- The intrinsic bonus is determined by comparing how similar the current observation is to the content of M. A larger difference results in a larger bonus.