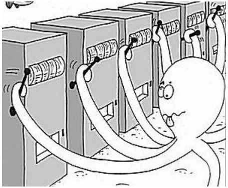
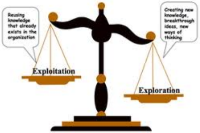
Reinforcement Learning

Exploration vs Exploitation

Marcello Restelli



March–April, 2015



Exploration vs Exploitation Dilemma

- **Online** decision making involves a fundamental choice:
  - **Exploitation**: make the **best decision** given current information
  - **Exploration**: gather **more information**
- The best long–term strategy may involve **short–term sacrifices**
- Gather **enough information** to make the best overall decisions

Examples

- Restaurant Selection
  - Exploitation: Go to favorite restaurant
  - Exploration: Try a new restaurant
- Online Banner Advertisements
  - Exploitation: Show the most successful advert
  - Exploration: Show a different advert
- Oil Drilling
  - Exploitation: Drill at the best known location
  - Exploration: Drill at a new location
- Game Playing
  - Exploitation: Play the move you believe is best
  - Exploration: Play an experimental move
- Clinical Trial
  - Exploitation: Choose the best treatment so far
  - Exploration: Try a new treatment

Common Approaches in RL

- ϵ**–Greedy**

$$a_t = \begin{cases} a_t^* & \text{with probability } 1-\epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases}$$

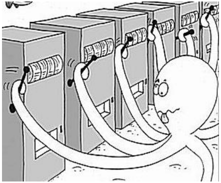
- **Softmax**
  - **Bias** exploration towards promising actions
  - Softmax action selection methods **grade action probabilities** by estimated values
  - The most common softmax uses a **Gibbs (or Boltzmann) distribution**:

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

  - τ is a **"computational" temperature**:
    - $\tau \to \infty$: $P = \frac{1}{|\mathcal{A}|}$
    - $\tau \to 0$: greedy

Multi–Arm Bandits (MABs)

- A multi–armed bandit is a tuple $\langle \mathcal{A}, R \rangle$
- $\mathcal{A}$ is a set of N possible **actions** (one per machine = arm)
- $R(r|a)$ is an **unknown** probability distribution of rewards given the action chosen
- At each time step t the agent **selects** an action $a_t \in \mathcal{A}$
- The environment generates a **reward** $r_t \sim R(\cdot, a)$
- The **goal** is to maximize cumulative reward: $\sum_{t=1}^{T} r_t$



Regret

- The **action–value** is the mean reward for action a

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

- The **optimal** value V^* is

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

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

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

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

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

- Maximize cumulative reward $\equiv$ minimize total regret

Counting Regret

- The count $N_t(a)$ is **expected number of selections** for action a
- The gap Δ_a is the **difference in value** between action a and optimal action a^*, $\Delta_a = V^* - Q(a)$
- Regret is a function of **gaps** and the **counts**

$$
\begin{aligned}
L_t &= \mathbb{E}\left[\sum_{t=1}^{T} V^* - Q(a_t)\right] \\
&= \sum_{a \in \mathcal{A}} \mathbb{E}[N_t(a)](V^* - Q(a)) \\
&= \sum_{a \in \mathcal{A}} \mathbb{E}[N_t(a)](\Delta_a)
\end{aligned}
$$

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

Greedy Algorithm

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

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t=1}^{T} r_t \mathbf{1}(a_t = a)$$

- The **greedy** algorithm selects action with **highest value**

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

- Greedy can lock onto a **suboptimal** action **forever**
- $\Rightarrow$ Greedy has **linear total regret**

ϵ–Greedy Algorithm

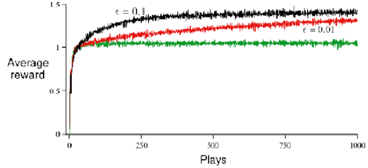
- The ϵ–greedy algorithm continues to **explore forever**
  - With probability $1 - \epsilon$ select $a = arg\max_{a \in \mathcal{A}} \hat{Q}(a)$
  - With probability ϵ select a random action
- Constant ϵ ensures **minimum regret**

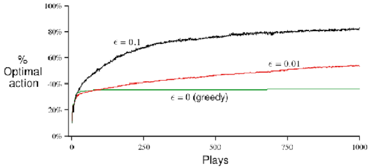
$$l_t \geq \frac{\epsilon}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \Delta_a$$

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

ϵ–Greedy on the 10–Armed Testbed

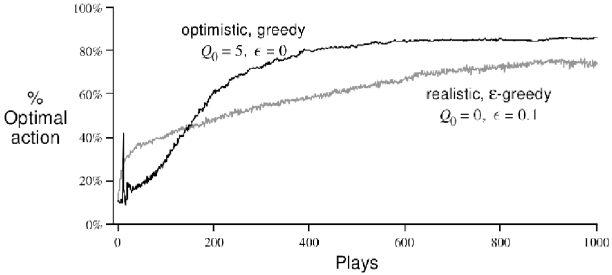
- $N = 10$ possible actions
- $Q(a)$ are chosen randomly from a normal distribution $\mathcal{N}(0, 1)$
- Rewards r_t are also normal $\mathcal{N}(Q(a_t), 1)$
- 1000 plays
- Results averaged over 2000 trials





Optimistic Initial Values

- **All** methods depend on $Q_0(a)$, i.e., they are **biased**
- Encourage exploration: **initialize** action values **optimistically**



Decaying ϵ_t–Greedy Algorithm

- Pick a **decay schedule** for $\epsilon_1, \epsilon_2, \ldots$
- Consider the following schedule

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

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

Two Formulations

- **Bayesian** formulation
  - $(Q(a_1), Q(a_2), \dots)$ are random variables with **prior** distribution $(f_1, f_2, \dots)$
  - **Policy**: choose an arm based on the priors $(f_1, f_2, \dots)$ and the observation history
  - Total **discounted** reward over an **infinite** horizon:

$$V^\pi(f_1, f_2, \dots) = \mathbb{E}_\pi \left\{ \sum_{t=0}^{\infty} \beta^t R^\pi(t) | f_1, f_2, \dots \right\} \quad (0 < \beta < 1)$$

- **Frequentist** formulation
  - $(Q(a_1), Q(a_2), \dots)$ are **unknown deterministic parameters**
  - **Policy**: choose an arm based on the observation history
  - **Total** reward over a **finite** horizon of length T

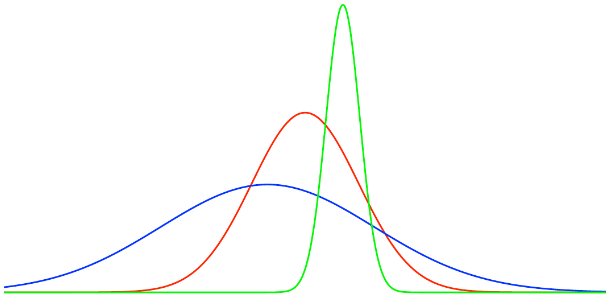
Bandit and MDP

- Multi-armed bandit as a class of **MDP**
  - N **independent** arms with **fully observable** states $[Z_1(t), \dots, Z_N(t)]$
  - **One** arm is activated at each time
  - Active arm changes state (**known Markov process**) and offers reward $R_i(Z_i(t))$
  - Passive arms are **frozen** and generate no reward
- Why is sampling stochastic processes with unknown distributions an MDP?
  - The state of each arm is the **posterior** distribution $f_i(t)$ (**information state**)
  - For an active arm, $f_i(t+1)$ is updated from $f_i(t)$ and the new observation
  - For a passive arm, $f_i(t+1) = f_i(t)$
- Solving MAB using DP: **exponential** complexity w.r.t. N

Gittins Index

- The index structure of the optimal policy [Gittins'74]
  - Assign each state of each arm a **priority** index
  - Activate the arm with highest current index value
- Complexity
  - Arms are **decoupled** (one N-dim to N separate 1-dim problems)
  - **Linear** complexity with N
  - **Forward** induction
  - Polynomial (cubic) with the state space size of a single arm
  - Computational cost is **too high** for **real-time** use
    - Approximations of the Gittins index
    - Thompson sampling
  - **Incomplete** learning

Optimism in Face of Uncertainty



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

Lower Bound

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

Theorem

Lai and Robbins Asymptotic total regret is at least ***logarithmic*** *in number of steps*

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

Upper Confidence Bounds

- Estimate an **upper confidence** $\hat{U}_t(a)$ for each action value
- Such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with **high probability**
- This depends on the **number of items** $N(a)$ has been selected
  - Small $N_t(a) \Rightarrow$ large $\hat{U}_t(a)$ (estimated value is **uncertain**)
  - Large $N_t(a) \Rightarrow$ small $\hat{U}_t(a)$ (estimated value is **accurate**)
- Select action maximizing **Upper Confidence Bound** (UCB)

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

Hoeffding's Inequality

Theorem

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

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

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

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

Calculating Upper Confidence Bounds

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

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

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

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

UCB1

This leads to the UCB1 algorithm

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

Theorem

At time T, the expected total regret of the UCB policy is at most

$$\mathbb{E}[L_T] \leq 8 \log T \sum_{a | \Delta_a < \Delta_{a^*}} \frac{1}{\Delta_a} + \left(1 + \frac{\pi^2}{3}\right) \sum_{a \in \mathcal{A}} \Delta_a$$

Example: UCB vs ϵ–Greedy on 10–armed Bandit

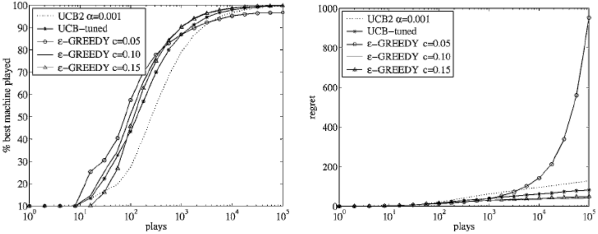


Figure 9. Comparison on distribution 11 (10 machines with parameters 0.9, 0.6, . . . , 0.6).

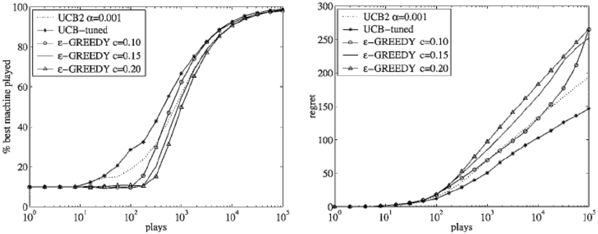


Figure 10. Comparison on distribution 12 (10 machines with parameters 0.9, 0.8, 0.8, 0.8, 0.7, 0.7, 0.7, 0.6, 0.6, 0.6).

Other Upper Confidence Bounds

Upper confidence bounds can be applied to other inequalities

- Bernstein's inequality
- Empirical Bernstein's inequality
- Chernoff inequality
- Azuma's inequality
- ...

The Adversarial Bandit Setting

- N arms
- At each round $t = 1, \dots, T$
  - The learner chooses $I_t \in 1, \dots, N$
  - At the same time the adversary selects reward vector $r_t = (r_{1,t}, \dots, r_{N,t}) \in [0, 1]^N$
  - The learner receives the reward $r_{I_t,t}$, while the rewards of the other arms are not received

Variation on Softmax
EXP3.P

- It is possible to drive regret down by **annealing** τ
- **EXP3.P**: Exponential weight algorithm for exploration and exploitation
- The probability of choosing arm a at time t is

$$\pi(a|t) = (1-\beta)\frac{w_t(a)}{\sum_{a' \in \mathcal{A}} w_t(a')} + \frac{\beta}{|\mathcal{A}|}$$

$$w_{t+1}(a) = \begin{cases} w_t(a)e^{\left(-\eta \frac{r_t(a)}{\pi_t(a)}\right)} & \text{if arm } a \text{ is pulled at } t \\ w_t(a) & \text{otherwise} \end{cases}$$

- $\eta > 0$ and $\beta > 0$ are the parameters of the algorithm
- **Regret**: $\mathbb{E}[L_T] \leq O(\sqrt{T|\mathcal{A}|\log|\mathcal{A}|})$

MAB with Infinitely Many Arms

- **Unstructured** set of actions
  - UCB Arm-Increasing Rule
- **Structured** set of actions
  - Linear Bandits
  - Lipschitz Bandits
  - Unimodal
  - Bandits in trees

The Contextual Bandit Setting

- At each round $t = 1, \ldots, T$
  - The world produces some context $s \in S$
  - The learner chooses $I_t \in 1, \ldots, N$
  - The world reacts with reward $R_{I_t,t}$
  - There is **no dynamics**
- Learn a good policy (low regret) for choosing actions **given context**
- Ideas
  - Run a different MAB for **each distinct context**
  - Perform **generalization** over contexts

Exploration–Exploitation in MDPs

- Multi–armed bandit are **one–step** decision–making problems
- MDPs represent **sequential** decision–making problems
- What exploration–exploitation approaches can be used in **MDPs**?
- How to **measure** the efficiency of an RL algorithm in formal terms?

Sample Complexity

- Let $\mathcal{M}$ be an **MDP** with N states, K actions, discount factor $\gamma \in [0, 1)$ and a maximal reward $R_{max} > 0$
- Let A be an **algorithm** that acts in the environment, producing experience: $s_0, a_0, r_1, s_1, a_1, r_2, \dots$
- Let $V_t^A = \mathbb{E}[\sum_{\tau=0}^{\infty} \gamma^\tau r_{t+\tau} | s_0, a_0, r_1, \dots, s_{t-1}, a_{t-1}, r_t, s_t]$
- Let V^* be the value function of the **optimal** policy

Definition [Kakade,2003]

Let $\epsilon > 0$ be a prescribed **accuracy** and $\delta > 0$ be an allowed **probability of failure.** The expression $\eta(\epsilon, \delta, N, K, \gamma, R_{max})$ is a **sample complexity** bound for algorithm A if independently of the choice of s_0, with probability at least $1 - \delta$, the number of timesteps such that $V_t^A < V^*(s_t) - \epsilon$ is at most $\eta(\epsilon, \delta, N, K, \gamma, R_{max})$.

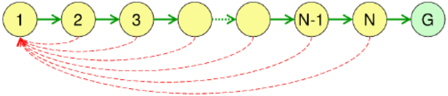
Efficient Exploration

Definition

An algorithm with sample complexity that is polynomial in $\frac{1}{\epsilon}, \log\frac{1}{\delta}, N, K, \frac{1}{1-\gamma}, R_{max}$ is called **PAC-MDP** (probably approximately correct in MDPs)

Exploration Strategies
PAC–MDP Efficiency

- ϵ**–greedy** and **Boltzmann** are not PAC-MDP efficient.
  - Their sample complexity is **exponential** in the number of states



  - Other approaches with exponential complexity:
    - **variance minimization** of action–values
    - **optimistic** value inizialization
    - state bonuses: frequency, recency, error, . . .
- Example of PAC–MDP efficient approaches:
  - model–based: E^3, R–MAX, MBIE
  - model–free: Delayed Q–learning
- **Bayesian RL**: optimal exploration strategy
  - only tractable for **very simple** problems

Explicit–Exploit–or–Explore (E^3) Algorithm

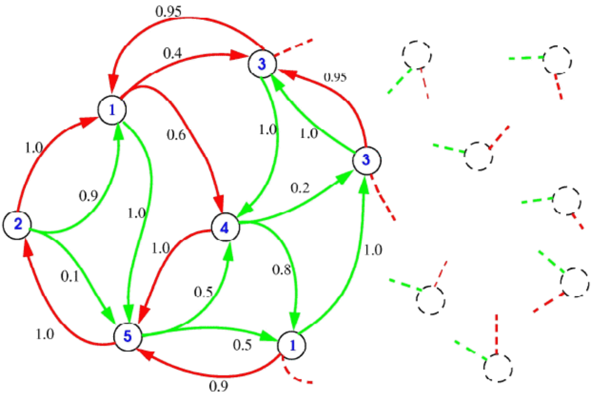
Kearns and Singh, 2002

- **Model–based** approach with polynomial sample complexity (PAC–MDP)
  - uses **optimism** in the face of uncertainty
  - assumes knowledge of **maximum reward**
- Maintains **counts for state and actions** to quantify confidence in model estimates
  - A state s is **known** if all actions in s have been sufficiently often executed
- From observed data, E^3 constructs **two** MDPs
  - MDP_{known}: includes known states with (approximately exact) estimates of P and R (drives **exploitation**)
  - $MDP_{unknown}$: MDP_{known} without reward + special state s' where the agent receives maximum reward (drives **exploration**)

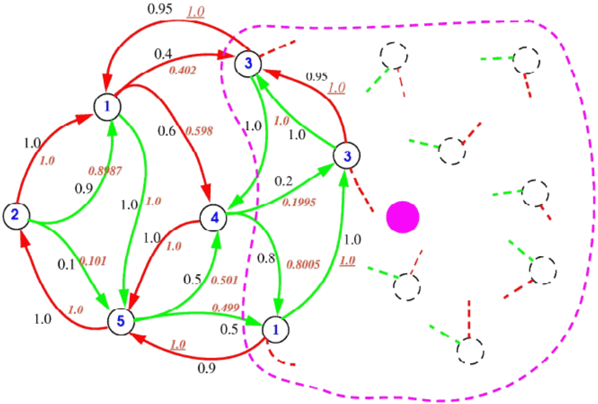
E^3 Sketch

Input: State s
Output: Action a
if s is **known then**
    Plan in MDP_{known}
    if resulting plan has value above some threshold **then**
        return first action of plan
    else
        Plan in $MDP_{unknown}$
        return first action of plan
    end if
else
    return action with the **least observations** in s
end if

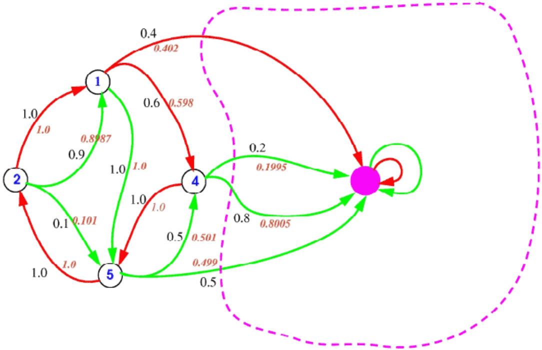
E^3 Example



E^3 Example


0.95
1.0
0.4
0.402
3
0.95
1.0
1
0.6
0.598
1.0
1.0
1.0
3
1.0
1.0
0.8987
0.9
0.2
0.1995
2
1.0
1.0
4
1.0
1.0
1.0
0.1
0.101
0.5
0.501
0.8
0.8005
1.0
1.0
0.499
1.0
5
0.5
1
1.0
1.0
0.9

E^3 Example



M: *true* known state MDP $\hat{M}$: *estimated* known state MDP

R–MAX

Brafman and Tennenholtz, 2002

- Similar to E^3: **implicit** instead of explicit exploration
- Based on reward function

$$R^{R-MAX}(s,a) = \begin{cases} R(s,a) & c(s,a) \geq m(s,a \text{ known}) \\ R_{max} & c(s,a) < m(s,a \text{ unknown}) \end{cases}$$

- **Sample Complexity**: $O\left(\frac{|\mathcal{S}|^2|\mathcal{A}|}{\epsilon^3(1-\gamma)^6}\right)$

Limitations of E^3 and R–MAX

- E^3 and R–MAX are called “efficient” because its sample complexity scales only **polynomially** in the **number of states**
- In natural environments, however, this number of states is **enormous**: it is **exponential** in the number of state variables
- Hence E^3 and R–MAX scale exponentially in the number of state variables
- **Generalization** over states and actions is crucial for exploration
  - **Relational Reinforcement Learning**: try to model structure of environments
  - **KWIK-R–MAX**: extension of R–MAX that is polynomial in the **number of parameters** used to approximate the state transition model

Delayed Q–learning

- **Optimistic** initialization
- **Greedy** action selection
- Q–learning updates in **batches** (**no** learning rate)
- Updates only if values **significantly decrease**
- Sample complexity: $O\left(\frac{|\mathcal{S}||\mathcal{A}|}{\epsilon^4(1-\gamma)^8}\right)$

Bayesian RL

- Agent maintains a distribution (**belief**) $b(m)$ over MDP models m.
    - typically, MDP structure is fixed; belief over the parameters
    - belief updated after each observation $(s, a, r, s') : b \to b'$
    - only tractable for very simple problems
- Bayes–optimal policy $\pi^* = arg\max_\pi V^\pi(b, s)$
    - no other policy leads to more rewards in expectation w.r.t. **prior distribution** over MDPs
    - solves the exploration–exploitation tradeoff implicitly: **minimizes uncertainty** about the parameters, while **exploiting** where it is certain
    - is **not** PAC-MDP efficient!