



Exporation and Exploitation

Nanxi Zhang

July 13, 2020





Outline:

Introduction to Reinforcement learning

Multi-arm Bandit Problem

Greedy Algorithm and ϵ -greedy Algorithm

UCB Algorithm

Bayesian approach and Thompson Sampling

Policy Gradient and Gradient Bandits

Summary



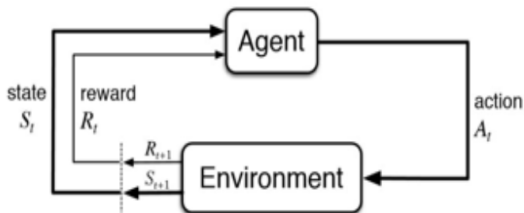
Motivation

- ▶ First, automation of repeated physical solutions
 - ▶ **Industrial revolution** (1750 - 1850) and Machine Age (1870 - 1940)
- ▶ Second, automation of repeated mental solutions
 - ▶ **Digital revolution** (1960 - now) and Information Age
- ▶ Next step: allow machines to find solutions themselves
 - ▶ **AI-revolution** (now-???)
- ▶ This requires learning autonomously how to make decisions



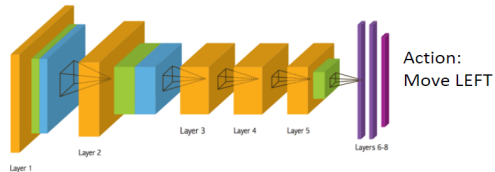
What is reinforcement learning?

- ▶ We, and other intelligent beings, learn by interacting with our environment
- ▶ This differs from certain other types of learning
- ▶ It is active rather than passive
- ▶ Interactions are often sequential—future interactions can depend on earlier ones
- ▶ We are goal-directed
- ▶ We can learn without examples of optimal behaviour



- ▶ Data are not i.i.d. Instead, a correlated time series data
- ▶ No instant feedback or label for correct action

Step3



Correct or Wrong???

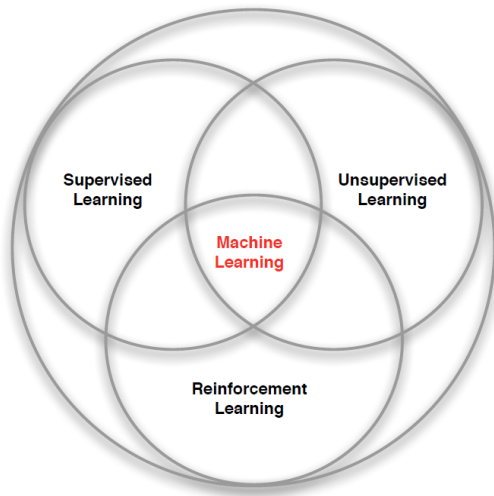
Don't know for now, until game is over

Delayed reward

5 / 34



Branches of machine learning





Core Concepts

- ▶ A reward is a scalar feedback signal
 - ▶ Indicate how well agent is doing at step t
 - ▶ Reinforcement Learning is based on the maximization of rewards: All goals of the agent can be described by the maximization of expected cumulative reward.
- ▶ RL agent may include one or more of these components.
 - ▶ **Agent state**(Constructing suitable state representation is not easy)
 - ▶ **Policy**: agent's behavior function, a map from state space \mathcal{S} to action space \mathcal{A} .
 - ▶ Stochastic Policy: $\pi(a|s) = P[A_t = a|S_t = s]$
 - ▶ Deterministic Policy: $a^* = \operatorname{argmax}_a \pi(a|s)$
 - ▶ **Value function(probably)**: how good is each state or action
 - ▶ **Model(optionally)**: agent's state representation of the environment. A model predicts what the environment will do next.(determines the transition matrix).







Multi-arm Bandit Problem: A simplified model

- ▶ Consider simple case: multiple actions, but only one state
- ▶ No sequential structure—past actions do not influence the future
- ▶ Formally: the distribution of R_t given A_t is identical and independent across time









Rat Example

	action	reward
Monday		
Tuesday		
Wednesday	?	





Rat Example

	action	reward
Monday		
Tuesday		
Wednesday		
Thursday	?	





Exploration & Exploitation

- ▶ Online decision-making involves a fundamental choice:
 - ▶ Performance based on current knowledge
 - ▶ Exploration: Increase knowledge
- ▶ The best long-term strategy may involve short-term sacrifices
- ▶ We want to gather enough information to make the best overall decisions



The Multi-armed Bandit

- ▶ We formalise the simplest setting
- ▶ \mathcal{A} is a known set of actions (or "arms")
- ▶ At each step t the agent selects an action $A_t \in \mathcal{A}$
- ▶ The environment generates a reward R_t
- ▶ The distribution $p(r|a)$ is fixed, but unknown
- ▶ The goal is to maximize cumulative reward $\sum_{i=1}^t R_i$
- ▶ Note: we sum over the whole lifetime of the agent
- ▶ Repeated 'game against nature'



Action Values

- ▶ The true action value for action a is the expected reward

$$q(a) = \mathbb{E} [R_t \mid A_t = a]$$

- ▶ A simple estimate is the average of the sampled rewards:

$$Q_t(a) = \frac{\sum_{n=1}^t R_n \mathcal{I}(A_n = a)}{\sum_{n=1}^t \mathcal{I}(A_n = a)}$$

where $\mathcal{I}(\text{True}) = 1$ and $\mathcal{I}(\text{False}) = 0$



Action Values

- ▶ This can be updated incrementally:

$$Q_t(A_t) = Q_{t-1}(A_t) + \alpha_t \underbrace{(R_t - Q_{t-1}(A_t))}_{\text{error}}$$

$$\forall a \neq A_t : Q_t(a) = Q_{t-1}(a)$$

with

$$\alpha_t = \frac{1}{N_t(A_t)}, \quad N_t(A_t) = N_{t-1}(A_t) + 1, \quad \text{and} \quad N_t(a) = 0, \forall a$$

- ▶ We can (and will, later) consider other **step sizes** α
- ▶ For instance, constant α would lead to **tracking**, rather than averaging



Regret

- ▶ The **optimal value** is

$$v_* = \max_{a \in \mathcal{A}} q(a) = \max_a \mathbb{E} [R_t \mid A_t = a]$$

- ▶ **Regret** is the opportunity loss for one step

$$v_* - q(A_t)$$

- ▶ In hindsight, I might 'regret' taking the tube rather than cycling
 - ▶ I might have regretted taking a bus even more
- ▶ The agent cannot observe, or even sample, the real regret directly
- ▶ But we can use it to analyze different learning algorithms



Regret

- ▶ Goal: Trade-off exploration and exploitation by minimizing total regret:

$$L_t = \sum_{i=1}^t (v_* - q(a_i))$$

- ▶ Maximise cumulative reward \equiv minimise total regret
- ▶ Note: the sum extends beyond (single step) episodes
- ▶ View extends over 'lifetime of learning', rather than over 'current episode'



Greedy Algorithm













The simplest action selection rule is to select the action (or one of the actions) with highest estimated action value, that is, to select at step t the action with:

$$A_t^* = \operatorname{argmax}_a Q_t(a)$$

- ▶ If the variance of the arm is 0, it's the best algorithm
- ▶ Spends no time exploring
- ▶ Regret is linear in t



Rat Example

t	A_t	R_t
1		
2		
3		
4		
5		
6		



- ▶ Regret can grow unbounded
- ▶ More interesting is how fast it grows
- ▶ The greedy policy has linear regret
- ▶ This means that, in expectation, the regret grows as a function that is linear in t
 - ▶ Suppose $p(\text{cheese}|\text{white}) = 0.1$ and $p(\text{cheese}|\text{black}) = 0.9$
 - ▶ Then $v^* = q(\text{black}) = 0.8$ and $q(\text{white}) = -0.8$
 - ▶ The greedy rat incurs regret of $1.6t$ (If the first two actions and rewards are as shown on the left)



ϵ —greedy

- ▶ We need to explore to learn the values
- ▶ One common solution: ϵ —greedy
- ▶ The ϵ —greedy algorithm continues to explore forever:
 - ▶ With probability $1 - \epsilon$ select $a = \operatorname{argmax}_a Q_t(a)$
 - ▶ With probability ϵ select a random action
- ▶ Will continue to select all suboptimal actions with, at least, probability $\epsilon/|\mathcal{A}|$
- ▶ With constant ϵ , has linear total regret



Lower Bound

Theorem(Lai and Robbins)

Asymptotic total regret is at least logarithmic in number of steps

$$\lim_{t \rightarrow \infty} L_t \geq \log t \sum_{a | \Delta_a > 0} \frac{\Delta_a}{KL(p(r | a) || p(r | a_*))}$$

Note: logarithmic is a whole lot better than linear!

- ▶ The performance of any algorithm is determined by similarity between optimal arm and other arms
- ▶ Hard problems have arms with similar distributions but different means
- ▶ This is described formally by the gap Δ_a and the similarity in distributions $KL(p(r | a) || p(r | a_*))$



UCB

The UCB algorithm:

$$a_t = \operatorname{argmax}_{a \in \mathcal{A}} Q_t(a) + c \sqrt{\frac{\log t}{N_t(a)}}$$

where c can be considered a hyper-parameter

Theorem

The UCB algorithm (with $c = \sqrt{2}$) achieves logarithmic expected total regret

$$L_t \leq 8 \sum_{a | \Delta_a > 0} \frac{\log t}{\Delta_a} + o\left(\sum_a \Delta_a\right)$$



Algorithm Idea

- ▶ ϵ -greedy action selection forces the non-greedy actions to be tried, but indiscriminately, with no preference for those that are nearly greedy or particularly uncertain.
- ▶ Select action maximizing upper confidence bound

$$a_t = \operatorname{argmax}_a Q_t(a) + U_t(a)$$

- ▶ The uncertainty depends on the number of times $N(a)$ has been selected
 - ▶ Small $N_t(a) \Rightarrow$ Large $U_t(a)$
 - ▶ Large $N_t(a) \Rightarrow$ small $U_t(a)$
 - ▶ For averages, the uncertainty decreases as $\sqrt{N_t(a)}$, by CLT
- ▶ Recall, we want to minimize $\sum_a N_t(a) \Delta_a$
 - ▶ If Δ_a is big, we want $N_t(a)$ to be small
 - ▶ If $N_t(a)$ is big, we want Δ_a to be small
 - ▶ Not all $N_t(a)$ can be small
 - ▶ What about Δ_a



Hoeffding's Inequality

Theorem

Let X_1, \dots, X_n be i.i.d. random variables in $[0, 1]$, and let $\bar{X}_t = \frac{1}{n} \sum_{i=1}^n X_i$ be the sample mean. Then

$$p(\mathbb{E}[X] \geq \bar{X}_n + u) \leq e^{-2nu^2}$$

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

$$e^{-2N_t(a)U_t(a)^2} = p$$

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

- ▶ Idea: reduce p as we observe more rewards, e.g. $p = 1/t$, $U_t(a) = \sqrt{\frac{\log t}{2N_t(a)}}$



UCB Upper Bound Proof



Bayesian Bandits

- ▶ Bayesian bandits model parameterized distributions over rewards, $p(R_t | \theta, a)$
- ▶ Compute posterior distribution over θ

$$p_t(\theta | a) \propto p(R_t | \theta, a) p_{t-1}(\theta | a)$$

- ▶ Allows us to inject rich prior knowledge
- ▶ Use posterior to guide exploration
 - ▶ Upper confidence bounds
 - ▶ Probability matching



Bayesian Bandits: Example

- ▶ Consider bandits with Bernoulli reward distribution: rewards are 0 or +1
- ▶ For each action, the prior could be a uniform distribution on $[0, 1]$
- ▶ This means we think each mean reward in $[0, 1]$ is equally likely
- ▶ The posterior is a Beta distribution $\text{Beta}(x_a, y_a)$ with initial parameters $x_a = 1$ and $y_a = 1$ for each action a
- ▶ Updating the posterior:
 - ▶ $x_{A_t} \leftarrow x_{A_t} + 1$ when $R_t = 0$
 - ▶ $y_{A_t} \leftarrow y_{A_t} + 1$ when $R_t = 1$



Probability Matching

- ▶ Probability matching selects action a according to probability that a is the optimal action

$$\pi_t(a) = p \left(q(a) = \max_{a'} q(a') \mid H_{t-1} \right)$$

- ▶ Probability matching is optimistic in the face of uncertainty: Uncertain actions have higher probability of being max
- ▶ Can be difficult to compute $\pi(a)$ analytically from posterior



Thompson Sampling

- ▶ Thompson Sampling:
 - ▶ Sample $Q_t(a) \sim p_t(q(a)), \forall a$
 - ▶ Select action maximising sample, $A_t = \operatorname{argmax}_{a \in \mathcal{A}} Q_t(a)$
- ▶ **Thompson sampling** is sample-based probability matching

$$\begin{aligned}\pi_t(a) &= \mathbb{E} \left[\mathcal{I} \left(Q_t(a) = \max_{a'} Q_t(a') \right) \right] \\ &= p \left(q(a) = \max_{a'} q(a') \right)\end{aligned}$$

- ▶ For Bernoulli bandits, Thompson sampling achieves Lai and Robbins lower bound on regret, and therefore is optimal



Upper Bound Proof for Two-armed Case



Policy Search

- ▶ What about learning policies $\pi(a)$ directly?
- ▶ For instance, define action preferences $H_t(a)$ and a policy

$$\pi(a) = \frac{e^{H_t(a)}}{\sum_b e^{H_t(b)}}$$

- ▶ The preferences do not have to have value semantics
- ▶ Instead, view them as learnable parameters
- ▶ Can we optimize the preferences ?



Policy Gradients

- ▶ Idea: update policy parameters such that the expected value increases
- ▶ We can consider gradient ascent on the expected value
- ▶ So, in the bandit case, we want to update:

$$\theta = \theta + \alpha \nabla_{\theta} \mathbb{E} [R_t \mid \theta]$$

where θ are the policy parameters

- ▶ Log-likelihood trick (also known as REINFORCE trick, Williams 1992):

$$\nabla_{\theta} \mathbb{E} [R_t \mid \theta] = \mathbb{E} [R_t \nabla_{\theta} \log \pi_{\theta} (A_t)]$$

- ▶ We can sample this!
- ▶ So

$$\theta = \theta + \alpha R_t \nabla_{\theta} \log \pi_{\theta} (A_t)$$



Gradient Bndits

- For soft max:

$$\begin{aligned} H_{t+1}(a) &= H_t(a) + \alpha R_t \frac{\partial \log \pi_t(A_t)}{\partial H_t(a)} \\ &= H_t(a) + \alpha R_t (\mathcal{I}(a = A_t) - \pi_t(a)) \end{aligned}$$

- \Rightarrow

$$\begin{aligned} H_{t+1}(A_t) &= H_t(A_t) + \alpha R_t (1 - \pi_t(A_t)) \\ H_{t+1}(a) &= H_t(a) - \alpha R_t \pi_t(a) \quad \text{if } a \neq A_t \end{aligned}$$

- Preferences for actions with higher rewards increase more (or decrease less), making them more likely to be selected again



Contents for today

- ▶ Brief introduction to RL
- ▶ MAB model setup
- ▶ Algorithms for MAB problems
 - ▶ Greedy & ϵ -Greedy
 - ▶ UCB
 - ▶ Thompson Sampling
 - ▶ Gradient Bandits



Not the end, yet...

- ▶ What can people in OR/OM do with RL?
- ▶ Revenue Management: Assortment Planning & Pricing
- ▶ Inventory: Design safe inventory policies
- ▶ Together with deep neural network, solve traditional hard problem, IP, for instance.
- ▶ ...