UNIT 2 MULTI-ARM BANDITS

Ami Munshi

Syllabus

2	**Multi-arm Bandits** An n-Armed Bandit Problem, Action-Value Methods, Incremental Implementation, Tracking a Non-stationary Problem, Optimistic Initial Values, Upper-Confidence-Bound	04

Course Outcomes

After completion of the course, students will be able to –

1. Apply the basics of Reinforcement Learning (RL) to compare with traditional control design
2. Correlate how RL relates and fits into the broader umbrella of machine learning, deep learning
3. Recommend value functions and appropriate algorithms for optimal decision-making
4. Design a dynamic programming approach to an industrial control problem

We will learn about

- N or k -arm bandit problem
- Exploration and Exploitation concept
- Greedy Algorithm
- Epsilon Greedy algorithm
- Optimistic Greedy
- Upper Confidence Bound

References

Text Books

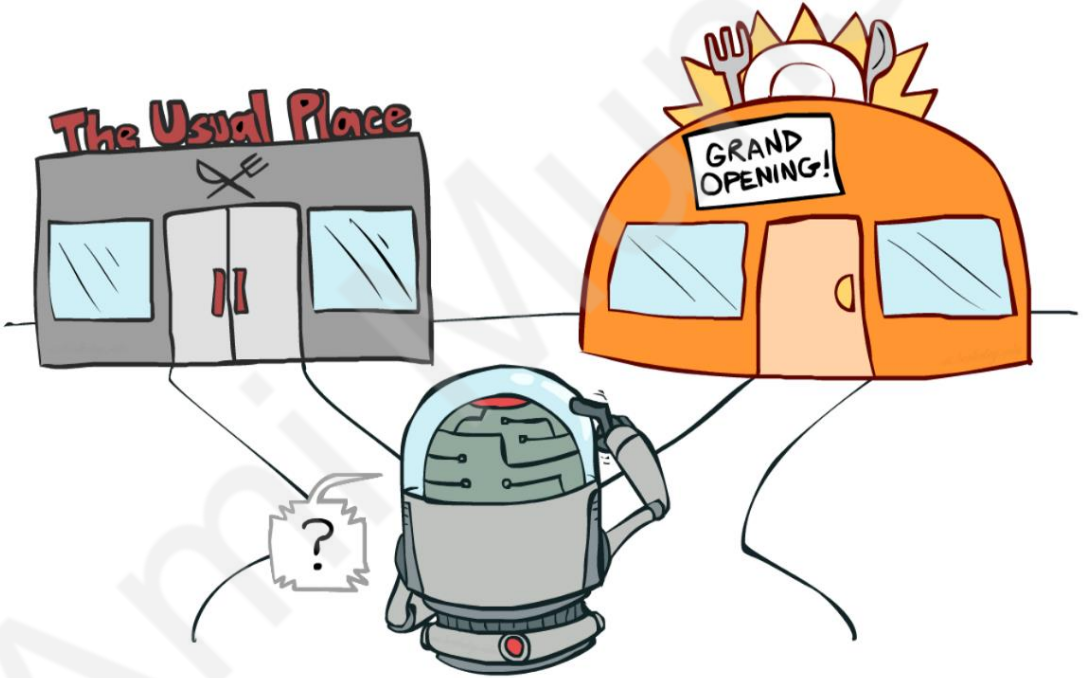
1. Laura Graesser and Wah Loon Keng, *Foundations of Deep Reinforcement Learning: Theory and Practice in Python*, 1st Edition, Pearson India/Padmavati Publisher, 2022.
2. Richard Sutton and Andrew Barto, *Reinforcement Learning: An Introduction*, 2nd Edition, MIT Press, 2018.
3. Abhishek Nandy and Manisha Biswas, *Reinforcement Learning: With Open AI, TensorFlow and Keras Using Python*, 1st Edition, Apress Publisher, 2017.

Reference Books

1. Nimish Sanghi, *Deep Reinforcement Learning with Python: With PyTorch, TensorFlow and OpenAI*, 2nd edition, Apress Publisher, 2021.
2. Alexander Zai and Brandon Brown, *Deep Reinforcement Learning in Action*, 1st Edition, Manning Publisher, 2020.
3. Csaba Szepesvari, *Algorithms for Reinforcement Learning*, 3rd Edition, Morgan & Claypool Publisher, 2019.

https://youtube.com/playlist?list=PLnn6VZp3hqNvRrdnMOVtgV64F_O-61C1D&feature=shared
https://youtube.com/playlist?list=PLsbSqisF2zAOFDbdDVOjjFvbHm6-p1udH&feature=shared
https://www.youtube.com/watch?v=gt-yR-ZGbbM&list=PLZ_sI4f41TGvthD8dA7daahlbLV0yDW0w&index=2

Exploration Vs Exploitation



Ref: http://ai.berkeley.edu/lecture_slides.html

Exploration Vs Exploitation-Examples

- Restaurant
  - Exploitation: Go to your favorite restaurant
  - Exploration: Try a new restaurant
- 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

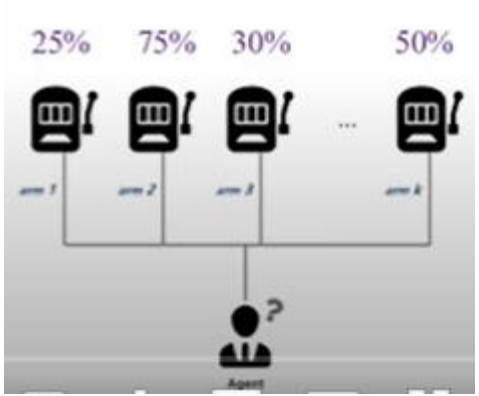
Ref: https://www.andrew.cmu.edu/course/10-403/slides/S19_lecture6_exploreexploitinbandits.pdf

What is the One Armed Bandit Problem?

- One-armed bandit is a slang term for slot machines
- One arm bandit is a type of slot machine with large metal pole on one side that you pull to make it work
- Spinning the one-armed bandit has some probability of winning
- We do not know probability distribution
- We can not know the probability distribution with limited number of trials

Ref: https://dictionary.cambridge.org/dictionary/english/one-armed-bandit
https://www.theaidream.com/post/multi-armed-bandit-problem
https://vectorportal.com/vector/slot-machine-vector-image.ai/22917

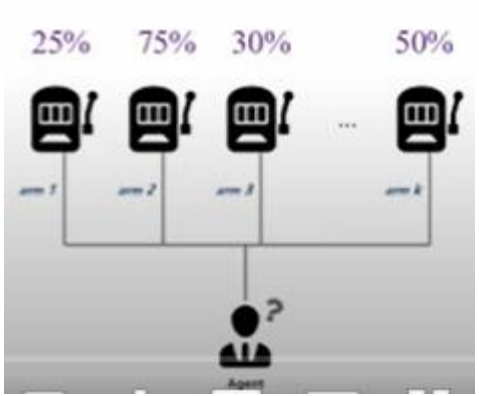
What is the k-Armed Bandit Problem?



- Named after the metaphor of a gambler facing multiple slot machines (bandits) and trying to maximize their rewards
- A classic problem in reinforcement learning and decision theory
- Involves a trade-off between exploration (trying new options) and exploitation (sticking with the best-known option)
- Every arm has a different probability distribution which is not known to the agent

Ref: https://www.youtube.com/watch?v=gt-yR-ZGbbM&list=PLZ_sI4f41TGvthD8dA7daahlbLV0yDW0w&index=2

What is the k-Armed Bandit Problem?



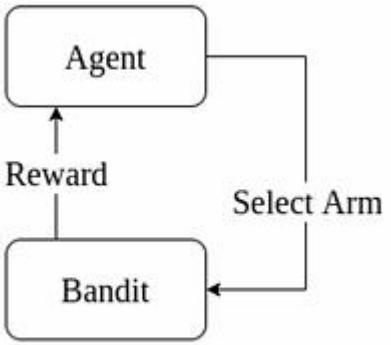
- Multi-Armed Bandit Problem (MAB) is a fundamental problem in the field of
  - reinforcement learning and decision-making under uncertainty
- Problem involves a gambler who has to choose among several slot machines (also called "one-armed bandits") with unknown payout probabilities
- Gambler's objective is to **maximize his or her total payout by choosing the best slot machine to play**

Ref: https://www.theaidream.com/post/multi-armed-bandit-problem

What is the k-Armed Bandit Problem?

- **Definition:**
  - An agent has to choose between N different actions (arms)
  - Each arm provides a reward from a probability distribution unique to that arm
  - Demonstrates classic problem of exploration vs exploitation
  - It is a one state problem
    - Rewards have no effect on future rewards
    - Probability distribution are stationary
- **Goal:**
  - Maximize the total reward over a series of trials (by doing cumulative actions)

k-armed bandit problem in RL scenario



k-Armed Bandit

Ref: https://www.theaidream.com/post/multi-armed-bandit-problem

Fun fact- Why is it called a multi-arm bandit problem

- A bandit is defined as someone who steals your money
- A one-armed bandit is a simple slot machine wherein you insert a coin into the machine, pull a lever, and get an immediate reward
- But why is it called a bandit?
- It turns out all casinos configure these slot machines in such a way that all **gamblers end up losing money!**

Exploration and Exploitation with reference to multi-arm bandit problem

- Exploitation and exploration are two key concepts in the context of the multi-armed bandit problem, which is a classic problem in reinforcement learning
- Goal of the problem is to repeatedly select one of several actions (known as "arms") in order to maximize the cumulative reward
- Exploitation
  - Refers to the strategy of selecting the arm with the highest estimated reward at each time step
  - Strategy is based on the assumption that the highest estimated reward is likely to be the true reward for that arm and
  - Selecting it repeatedly will result in the highest cumulative reward over time
- **Exploitation is focused on exploiting the current knowledge of the reward distribution and maximizing the short-term reward**

Ref: https://www.theaidream.com/post/multi-armed-bandit-problem

Exploration and Exploitation with reference to multi-arm bandit problem

- **Exploration**
  - Refers to the strategy of selecting arms at random or selecting arms with low estimated reward in order to gather more information about their true reward distribution
  - Strategy is based on the assumption that the estimated reward distribution may be incorrect and
  - Gathering more information about the reward distribution of each arm will lead to better long-term performance
- **Exploration is focused on exploring the unknown aspects of the reward distribution in order to maximize the long-term reward**

Ref: https://www.theaidream.com/post/multi-armed-bandit-problem

Exploration and Exploitation with reference to multi-arm bandit problem

- Balancing exploitation and exploration is a key challenge in the multi-armed bandit problem
- If the agent focuses too much on exploitation, it may get stuck on a sub-optimal arm and miss out on the best arm
- Conversely, if the agent focuses too much on exploration, it may waste time on low-reward arms and miss out on the cumulative reward from the best arm
- In summary
  - Exploitation and exploration are two competing strategies for selecting arms in the multi-armed bandit problem
  - **Balancing these strategies is essential to achieve the best overall performance**

Ref: https://www.theaidream.com/post/multi-armed-bandit-problem

k-Armed Bandit Problem

- In k-armed bandit problem
  - Each of the k actions has an expected or mean reward given that that action is selected
  - Let us call this the *value* of that action
  - We denote the action selected on time step t as A_t,
  - And the corresponding reward as R_t
- The value then of an arbitrary action a, denoted $q_*(a)$, is the **expected reward** given that a is selected
  - $q_*(a) = \mathrm{E}[(R_t | A_t = a)]$

Ref: Richard Sutton, Andrew Barto, "Reinforcement Learning: An Introduction", 2nd edition, MIT Press, 2020

k-Armed Bandit Problem

- The value then of an arbitrary action a, denoted $q_*(a)$, is the **expected reward** given that a is selected
  - $q_*(a) = \mathrm{E}[(R_t|A_t = a)]$
- If you knew the value of each action, then it would be trivial to solve the k-armed bandit problem:
  - you would always select the action with highest value
- We assume that you do not know the action values with certainty, although you may have estimates
- We denote the **estimated value** of action a at time step t as $Q_t(a)$
- **We would like $Q_t(a)$ to be close to $q_*(a)$**

Ref: Richard Sutton, Andrew Barto, "Reinforcement Learning: An Introduction", 2nd edition, MIT Press, 2020

Knowing the meaning of Expected Value and Estimated Value

- **Expected Value**
  - The expected value (or true value or mean value) of an action (arm) is the long-run average reward that would be obtained if that action were chosen repeatedly an infinite number of times
  - It is a theoretical measure based on the underlying probability distribution of rewards for that action
  - Mathematically, for action a, the expected value is
    - $q_*(a) = \mathrm{E}[(R_t|A_t = a)] = \sum r.P(r|a)$

Knowing the meaning of Expected Value and Estimated Value

- **Estimated Value**
  - The estimated value (or sample mean) of an action is the average reward obtained from actually selecting that action a certain number of times
  - It is an empirical measure based on observed rewards
  - Mathematically, for action a that has been chosen n_a times, with rewards $r_1, r_2, .. r_{n_a}$ the estimated value is given by
  - $Q(a) = \left(\frac{1}{n_a}\right) \sum_{i=1}^{n_a} r_i$
- As more rewards are observed by selecting the action more times, the estimated value becomes a better approximation of the expected value

k-Armed Bandit Problem

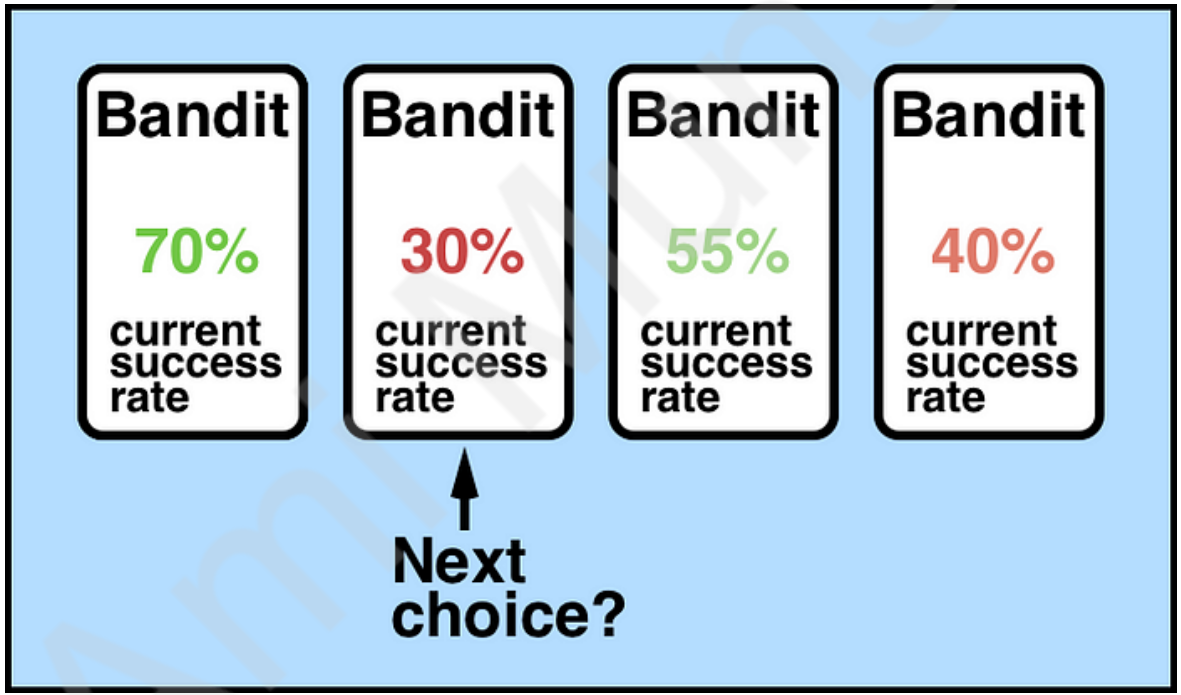
- If you maintain estimates of the action values, then at any time step there is at least one action whose estimated value is greatest
- We call these the greedy actions
- When you select one of these actions, we say that you are exploiting your current knowledge of the values of the actions
- If instead you select one of the non-greedy actions, then we say you are exploring, because this enables you to improve your estimate of the non-greedy action's value
- Exploitation is the right thing to do to maximize the expected reward on the one step
- But exploration may produce the greater total reward in the long run

Ref: Richard Sutton, Andrew Barto, "Reinforcement Learning: An Introduction", 2nd edition, MIT Press, 2020

Conflict between exploration and exploitation and when to select what

- For example, suppose a greedy action's value is known with certainty, while several other actions are estimated to be nearly as good but with substantial uncertainty
- Uncertainty is such that at least one of these other actions probably is actually better than the greedy action, but you don't know which one
- If you have many time steps ahead on which to make action selections, then it may be better to explore the non-greedy actions and discover which of them are better than the greedy action
- Reward is lower in the short run, during exploration, but higher in the long run because after you have discovered the better actions, you can exploit them many times
- **Because it is not possible both to explore and to exploit with any single action selection, one often refers to the "conflict" between exploration and exploitation**
- In any specific case, whether it is better to explore or exploit depends in a complex way on the precise values of the estimates, uncertainties, and the number of remaining steps

Example of Bernoulli Multi armed bandit

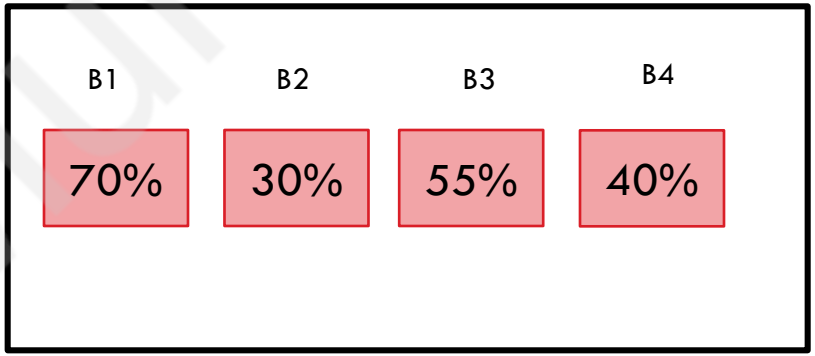


Ref: https://towardsdatascience.com/solving-the-multi-armed-bandit-problem-b72de40db97c

Simplest approach or Naïve Approach- Exploit only approach

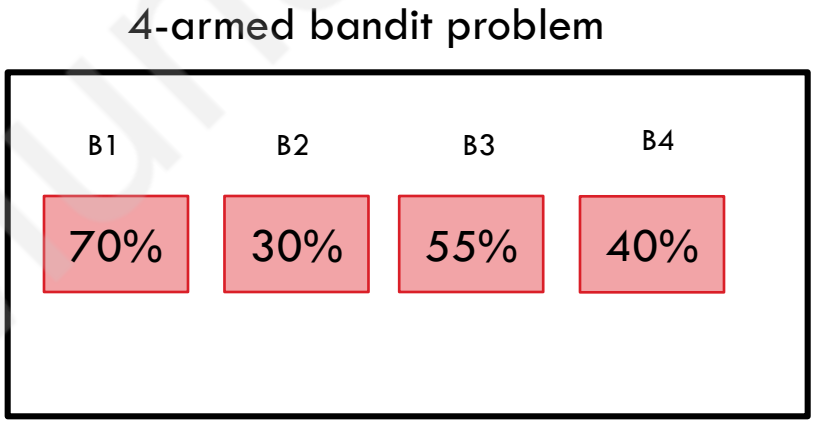
- Playing with one machine for many rounds
- Here the agent does not know the probability of winning for each slot
- Let us assume that the agent keeps playing on machine B2
- This is exploitation action
- After $t = 1000$ steps, the expected reward would be
  - 1000*0.3= 300
- Maximum reward that the agent could have obtained if probability of winning was known for each machine would be
  - 1000*0.7= 700
- Regret in this case is
  - 700-300=400

4-armed bandit problem



Simplest approach or Naïve Approach- Explore only approach

- Playing with each machine for equal rounds
- Here the agent does not know the probability of winning for each slot
- Let us assume that the agent plays on each machine for 250 rounds
- This is exploration action
- After $t = 1000$ steps, the expected reward would be
  - 250*0.7+250*0.3+250*0.55+250*0.4=487.5
- Maximum reward that the agent could have obtained if probability of winning was known for each machine would be
  - 1000*0.7= 700
- Regret in this case is
  - 700-487.5=212.5



Action-value methods/Value function based methods

- Methods for
  - estimating the values of actions and
  - for using the estimates to make action selection decisions
- **Collectively these are called action-value methods**
- **We don't know the $q_*(a)$. Hence will estimate $Q_*(a)$**
  - **Q is estimate of q**

Action-value method

- We know that the true value of an action is the mean reward when that action is selected
- One natural way to estimate this is by averaging the rewards actually received

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}},$$

- Where $\mathbb{I}$ predicate denotes the random variable that is 1 if predicate is true and 0 if it is not
- If the denominator is zero, then we instead define $Q_t(a)$ as some default value, such as 0
- As the denominator goes to infinity, by the law of large numbers, $Q_t(a)$ converges to $q_*(a)$
- Note: Refer this link to understand law of large numbers- https://en.wikipedia.org/wiki/Law_of_large_numbers

Action-value method

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}},$$

- We call this the sample-average method for estimating action values because each estimate is an average of the sample of relevant rewards
- Of course this is just one way to estimate action values, and not necessarily the best one
- Nevertheless, for now let us stay with this simple estimation method and turn to the question of how the estimates might be used to select actions

Greedy Action Selection method

- Simplest action selection rule is to select one of the actions with the highest estimated value, that is, one of the greedy actions as defined in the previous section
- If there is more than one greedy action, then a selection is made among them in some arbitrary way, perhaps randomly
- We write this greedy action selection method as

$$A_t \doteq \arg\max_a Q_t(a)$$

- where $argmax_a$ denotes the action a for which the expression that follows is maximized (with ties broken arbitrarily)
- Greedy action selection always exploits current knowledge to maximize immediate reward
- It spends no time at all sampling apparently inferior actions to see if they might really be better

Ref: Richard Sutton, Andrew Barto, "Reinforcement Learning: An Introduction", 2nd edition, MIT Press, 2020

Fixed Exploration +Greedy

- Allocate fixed time steps for exploration where the bandits are tried uniformly at random
- Estimate rewards for all the actions using
  - $Q_t(a) = \left(\frac{1}{n_a}\right)\sum_{i=1}^{n_a} r_i$ or $\quad Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}},$
- Select the action that is optimal for the estimated mean rewards given all data using

$$A_t \doteq \arg\max_a Q_t(a)$$

- If there is a tie in optimal actions, select randomly from the actions that are optimal

$\in$ − Greedy

- A simple alternative
  - Behave greedily most of the time (exploit)
  - But every once in a while, say with small probability $\in$, instead select randomly from among all the actions with equal probability (explore), independently of the action-value estimates
- We call method using this near-greedy action selection rule $\in$ − greedy method
- An advantage of these methods is that,
  - In the limit as the number of steps increases, every action will be sampled an infinite number of times, thus ensuring that all the $Q_t(a)$ converge to their respective $\mathrm{q}_*(a)$

$\in$ – Greedy policy concept

- Algorithm takes
  - the best action (exploit) or
  - Chooses the random action (explore)
- Steps to do this
  - Select $\in$ value between [0,1]
  - Generate a random number x between [0,1]
    - If $x > \in$ then choose greedy action (exploit)
      - That is action for which $A_t \doteq \arg\max_a Q_t(a)$
      -
    - If $x \leq \in$ then choose random action (explore)

$\in$ – Greedy policy concept

- In case of greedy action (exploit)-
- Action value can be estimated using past experience using

- $Q_t(a) = \left(\frac{1}{n_a}\right) \sum_{i=1}^{n_a} r_i$ or $Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}},$

- The best action at time t is selected using $A_t \doteq \arg\max_a Q_t(a)$

$\in$ – Greedy policy concept

- If $x > \in$ then choose greedy action (exploit)
- If $x \leq \in$ then choose random action (explore)
- Example
  - Let number of steps=1000
  - Let us select $\in$=0.2
  - Then for 0.2*1000=200 times , we ignore the information available and randomly select an action (explore)
  - For 0.8*1000 =800 times greedy action will be selected based on the available knowledge (exploit)

Think

- What happens if $\epsilon=0$?
  - Exploit only
  - Explore only

Optimistic Greedy Action Selection method

- This approach is named as such because unlike the Epsilon-Greedy Algorithm, we start off by being optimistic about the win rates of the bandits and assign a higher initial win rate
- Sometimes even more than what is actually possible
- As for the exploration factor, instead of being reliant on Epsilon we rely on the fact that with each iteration the optimistic initial win rates will keep falling towards their actual win rates
- The focus is on simply choosing the bandit with highest estimated win rate during this calibration process
- This fall in win rate estimates would reach a point where all sub-optimal bandit's win rate estimate would fall below the win rate estimate of the most optimal bandit

Ref: https://blog.devgenius.io/multi-armed-bandit-problem-optimistic-initial-values-89c10f04974c

Example for optimistic greedy algorithm

We have two slot machines (single arm bandits) Machine A & Machine B with actual win rates of 25% and 75% respectively. The actual win rates are unknown to the model. Optimistic-greedy algorithm is used to select the optimal bandit. Assume that we start with an optimistic initial estimate of 200% win rate for each of the machines, even though the highest win rate that is ever possible is 100%. Fill up the table below with $M1_n$, $M2_n$ values and reason why M1 or M2 was chosen at each iteration:

Time Step Number (n)	Machine 1 (M1)	Win from Machine 1	Machine 2 (M2)	Win from Machine 2	Reason why M1 or M2 was chosen?
0	$M1_0 = 200\%$	-	$M2_0 = 200\%$	-	
1	$M1_1 = ?$	No	$M2_1 = ?$	-	?
2	$M1_2 = ?$	-	$M2_2 = ?$	Yes	?
3	$M1_3 = ?$	-	$M2_3 = ?$	No	?
4	$M1_4 = ?$	No	$M2_4 = ?$	-	?
5	$M1_5 = ?$	-	$M2_5 = ?$	No	?

What do you expect will happen after Iteration 5 in the context of this problem?

Ref: https://blog.devgenius.io/multi-armed-bandit-problem-optimistic-initial-values-89c10f04974c

Note

- Greedy Action-Selection is a special case of Epsilon-Greedy with Epsilon = 0
- Random Action Selection is a special case of Epsilon-Greedy with Epsilon = 1

Example

Exercise 2.1 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? □

Ref: Richard Sutton, Andrew Barto, "Reinforcement Learning: An Introduction", 2nd edition, MIT Press, 2020

Example

Exercise 2.1 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? □

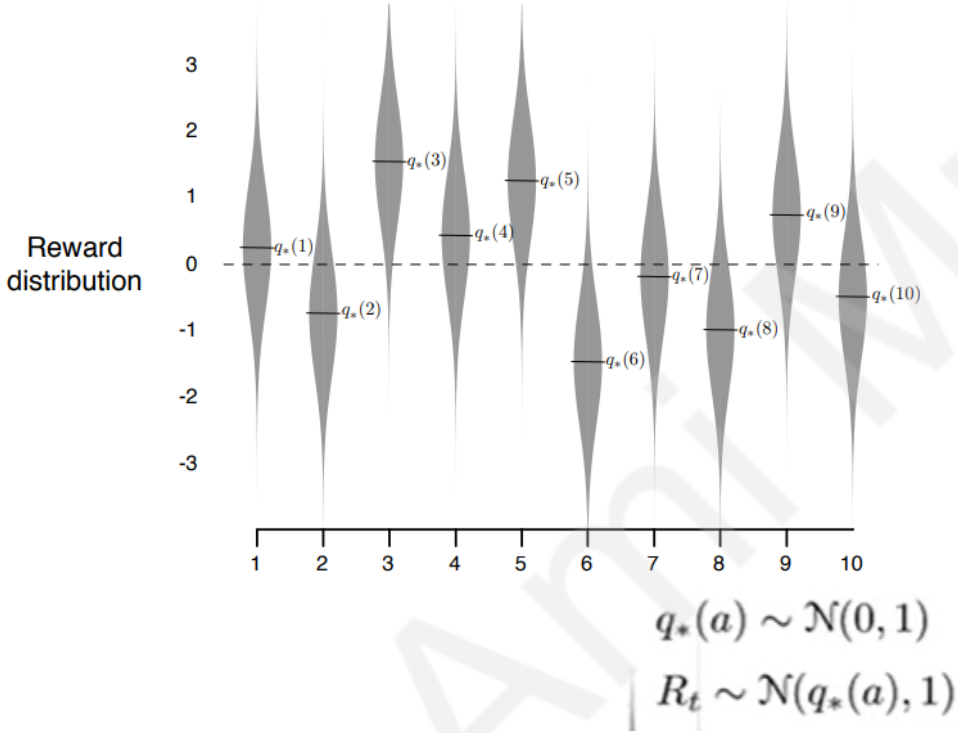
- According to $\in$ − Greedy algorithm,
  - $A = \begin{cases} argmax_a Q(a) \; with \; probability \; 1 - \in \\ a \; random \; action \; with \; probability \; \in \end{cases}$
- Hence probability that the greedy action is selected is $1 - \in$=1-0.5=0.5

Ref: Richard Sutton, Andrew Barto, "Reinforcement Learning: An Introduction", 2nd edition, MIT Press, 2020

10-armed test bed

- To roughly assess the relative effectiveness of the greedy and ϵ-greedy action-value methods, these are compared numerically on a suite of test problems
- This was a set of 2000 randomly generated k-armed bandit problems with $k = 10$
- For each bandit problem, such as the one shown in the figure

10-armed test bed



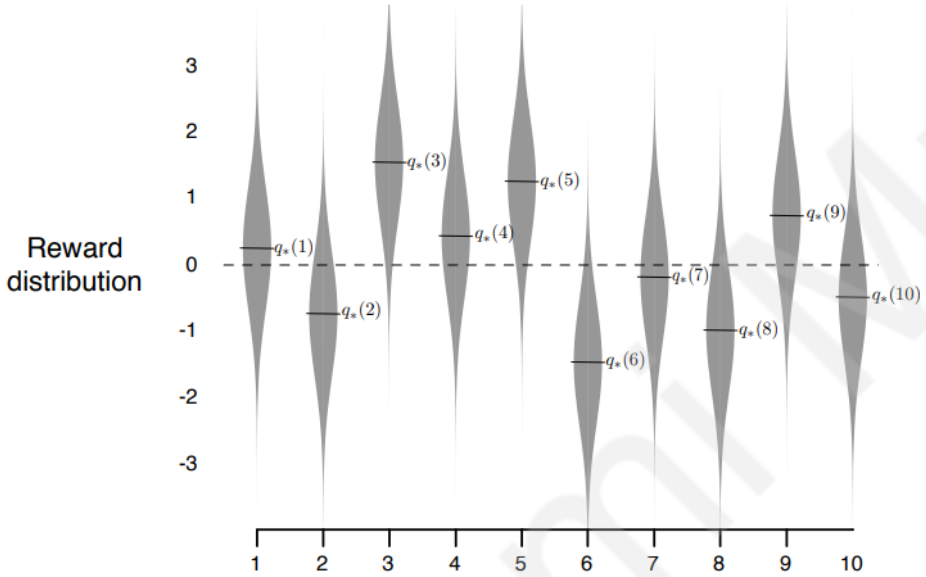
$$q_*(a) \sim \mathcal{N}(0, 1)$$

$$R_t \sim \mathcal{N}(q_*(a), 1)$$

- The action values, $q_*(a), a = 1,2 \ldots 10$, were selected according to a normal (Gaussian) distribution with mean 0 and variance 1
- Then, when a learning method applied to that problem selected action A_t at time step t, the actual reward, R_t, was selected from a normal distribution with mean $q_*(a)$and variance 1
- These distributions are shown in gray in the figure
- This suite of test tasks is called the 10-armed testbed

Ref: Richard Sutton, Andrew Barto, "Reinforcement Learning: An Introduction", 2nd edition, MIT Press, 2020

10-armed test bed



- For any learning method, we can measure its performance and behavior as it improves with experience over 1000 time steps when applied to one of the bandit problems
- This makes up one run
- Repeating this for 2000 independent runs, each with a different bandit problem, we obtained measures of the learning algorithm's average behavior

Ref: Richard Sutton, Andrew Barto, "Reinforcement Learning: An Introduction", 2nd edition, MIT Press, 2020

10-armed test bed

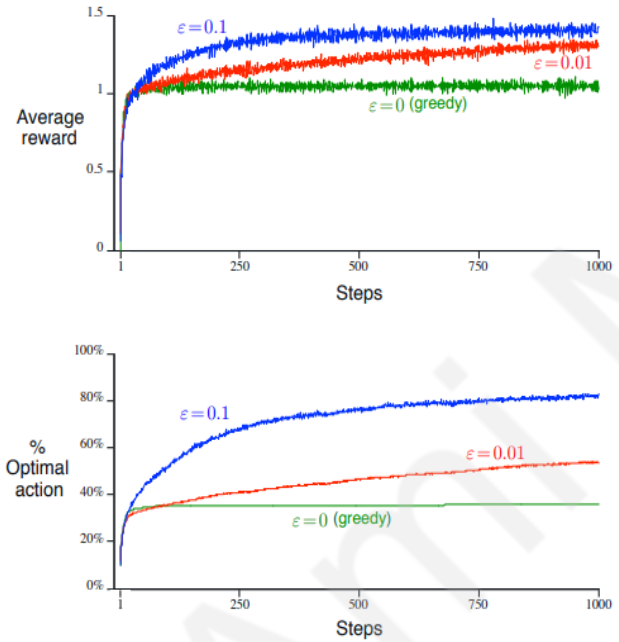


Figure 2.2: Average performance of ε-greedy action-value methods on the 10-armed testbed. These data are averages over 2000 runs with different bandit problems. All methods used sample averages as their action-value estimates.

- Figure compares a greedy method with two " $\in$ -greedy methods ($\in = 0.01$ and $\in = 0.1$), as described above, on the 10-armed testbed
- All the methods formed their action-value estimates using the sample-average technique (with an initial estimate of 0)
- The upper graph shows the increase in expected reward with experience
- The greedy method improved slightly faster than the other methods at the very beginning, but then leveled off at a lower level
- It achieved a reward-per-step of only about 1, compared with the best possible of about 1.54 on this testbed.
- The greedy method performed significantly worse in the long run because it often got stuck performing suboptimal actions

10-armed test bed

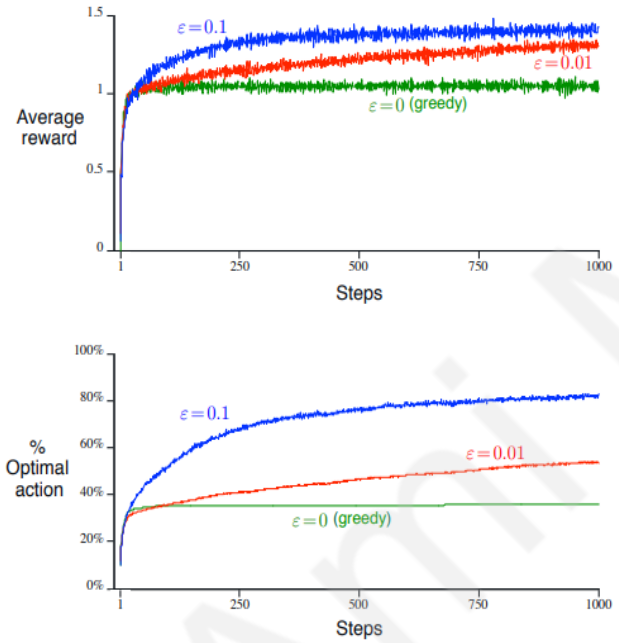


Figure 2.2: Average performance of ε-greedy action-value methods on the 10-armed testbed. These data are averages over 2000 runs with different bandit problems. All methods used sample averages as their action-value estimates.

- The lower graph shows that the greedy method found the optimal action in only approximately one-third of the tasks
- In the other two-thirds, its initial samples of the optimal action were disappointing, and it never returned to it
- The ∈ -greedy methods eventually performed better because they continued to explore and to improve their chances of recognizing the optimal action

Home work

- Some observations about $\epsilon = 0.01$ and $\epsilon = 0.1$
- "Advantage of ϵ -greedy over greedy methods depends on the task". Justify
  - Refer to Pg number 30 of this book
  - Richard Sutton, Andrew Barto, "Reinforcement Learning: An Introduction", 2nd edition, MIT Press, 2020

Example on ∈ Greedy

Exercise 2.2: Bandit example Consider a k-armed bandit problem with $k = 4$ actions, denoted 1, 2, 3, and 4. Consider applying to this problem a bandit algorithm using ε-greedy action selection, sample-average action-value estimates, and initial estimates of $Q_1(a) = 0$, for all a. Suppose the initial sequence of actions and rewards is $A_1 = 1$, $R_1 = -1$, $A_2 = 2$, $R_2 = 1$, $A_3 = 2$, $R_3 = -2$, $A_4 = 2$, $R_4 = 2$, $A_5 = 3$, $R_5 = 0$. On some of these time steps the ε case may have occurred, causing an action to be selected at random. On which time steps did this definitely occur? On which time steps could this possibly have occurred? □

Incremental Implementation

- Action-value methods we have discussed so far all estimate action values as sample averages of observed rewards
- How these averages can be computed in a computationally efficient manner?
- To simplify notation we concentrate on a single action. Let R_i now denote the reward received after the i_{th} selection of this action, and
- Let Q_n denote the estimate of its action value after it has been selected $n-1$ times, which we can now simply write as
  - $Q_n = \frac{R_1+R_2+..R_{n-1}}{n-1}$
- The obvious implementation would be to maintain a record of all the rewards and then perform this computation whenever the estimated value was needed
- However, if this is done, then the memory and computational requirements would grow over time as more rewards are seen

Incremental Implementation

$$Q_n = \frac{R_1 + R_2 + .. R_{n-1}}{n-1}$$

- This implementation would be to maintain a record of all the rewards and then perform this computation whenever the estimated value was needed
- Limitations
  - Memory and computational requirements would grow over time as more rewards are seen
  - Each additional reward would require additional memory to store it and additional computation to compute the sum in the numerator

Incremental Implementation

- It is easy to devise incremental formulas for updating averages with small, constant computation required to process each new reward
- Given Q_n and the n_{th} reward, R_n, the new average of all n rewards can be computed by

Incremental Implementation

$$
\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\Big(R_n + (n-1)Q_n\Big) \\
&= \frac{1}{n}\Big(R_n + nQ_n - Q_n\Big) \\
&= Q_n + \frac{1}{n}\Big[R_n - Q_n\Big],
\end{aligned}
$$

- which holds even for n = 1, obtaining Q2 = R1 for arbitrary Q1. This implementation requires memory only for Qn and n, and only the small computation for each new reward

Incremental Implementation

- $Q_{n+1} = Q_n + \frac{1}{n}(R_n - Q_n)$
- The general form is

$$NewEstimate \leftarrow OldEstimate + StepSize\left[Target - OldEstimate\right]$$

Error

- Error is reduced by taking a step toward the "Target."
- Target is presumed to indicate a desirable direction in which to move, though it may be noisy
- In the case above, for example, the target is the n_{th} reward
- Step-size parameter $1/n$ is denoted by α *or* $\alpha_t(a)$

Pseudo code for ∈-greedy action selection

- Pseudocode for a complete bandit algorithm using incrementally computed sample averages and ∈-greedy action selection is shown in the box below

A simple bandit algorithm

Initialize, for $a = 1$ to k:
$Q(a) \leftarrow 0$
$N(a) \leftarrow 0$

Loop forever:
$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$
$R \leftarrow bandit(A)$
$N(A) \leftarrow N(A) + 1$
$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\left[R - Q(A)\right]$

Tracking a non stationary problem

- Sample average method used in stationary problems to estimate the value is given by $Q_{n+1} = Q_n + \frac{1}{n}(R_n - Q_n)$
- Averaging methods discussed so far are appropriate for stationary bandit problems, that is, for bandit problems in which the reward probabilities do not change over time
- As noted earlier, we often encounter reinforcement learning problems that are effectively nonstationary
- Any examples of non stationary reward function??
- In such cases it makes sense to **give more weight to recent rewards** than to long-past rewards
- One of the most popular ways of doing this is to **use a constant step-size parameter**

Tracking a non stationary problem

- For example, the incremental update rule for updating an average Q_n of the $n-1$ rewards is modified as
  - $Q_{n+1} = Q_n + \alpha\,(R_n - Q_n)$
- where the step-size parameter $\alpha \epsilon (0,1]$ is constant

Tracking a non stationary problem

$$
\begin{aligned}
Q_{n+1} &= Q_n + \alpha\Big[R_n - Q_n\Big] \\
&= \alpha R_n + (1-\alpha)Q_n \\
&= \alpha R_n + (1-\alpha)\left[\alpha R_{n-1} + (1-\alpha)Q_{n-1}\right] \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-2} + \\
&\qquad \cdots + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i.
\end{aligned}
$$

- This results in Q_{n+1} being a weighted average of past rewards and the initial estimate Q_1

Tracking a non stationary problem

- We call this a weighted average because the sum of the weights is $(1-\alpha)^n + \sum_{i=1}^{\alpha} \alpha(1-\alpha)^{n-i} = 1$
- Note that the weight, $\alpha(1-\alpha)^{n-i}$ given to reward R_i, depends on how many rewards ago, $n - i =$, it was observed
- Quantity $1-\alpha$ is less than 1, thus weight given to R_i decreases as number of intervening rewards increases
- Weight decays exponentially according to exponent on $1-\alpha$
- Hence this is called as exponentially recency weighted average

Optimistic initial values

- Initial action values can also be used as a simple way to encourage exploration
- Suppose that instead of setting the initial action values to zero, as we did in the 10-armed testbed, we set them all to $+5$
- Recall that the $q_*(a)$in this problem are selected from a normal distribution with mean 0 and variance 1
- An initial estimate of $+5$ is thus wildly optimistic
- But this optimism encourages action-value methods to explore
- Whichever actions are initially selected, the reward is less than the starting estimates; the learner switches to other actions, being "disappointed" with the rewards it is receiving
- The result is that all actions are tried several times before the value estimates converge
- The system does a fair amount of exploration even if greedy actions are selected all the time

Optimistic initial values

- Fig shows the performance on the 10-armed bandit testbed of a greedy method using Q1(a) = +5, for all a
- For comparison, also shown is an ∈-greedy method with Q1(a) = 0
- Initially, the optimistic method performs worse because it explores more, but eventually it performs better because its exploration decreases with time
- We call this technique for encouraging exploration optimistic initial values
- This that can be quite effective on stationary problems

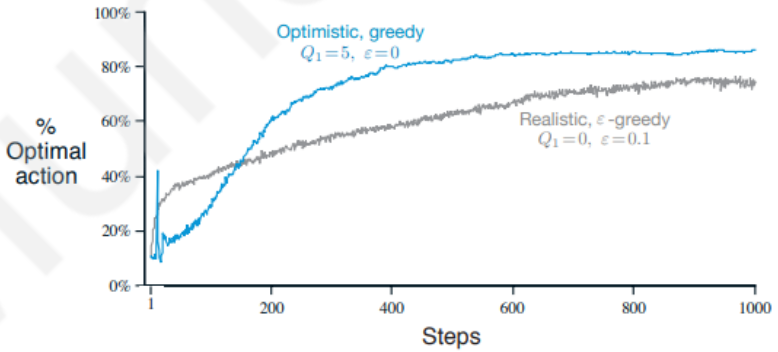


Figure 2.3: The effect of optimistic initial action-value estimates on the 10-armed testbed. Both methods used a constant step-size parameter, $\alpha = 0.1$.

Upper Confidence Bound

- Exploration is needed because there is always uncertainty about the accuracy of the action-value estimates
- Greedy actions are those that look best at present, but some of the other actions may actually be better
- 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
- It would be better to select among the non-greedy actions according to their potential for actually being optimal, taking into account both
  - how close their estimates are to being maximal and
  - the uncertainties in those estimates
- One effective way of doing this is to select actions according to

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

Upper Confidence Bound

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

- where $\log_e t$ denotes the natural logarithm of t
- $N_t(a)$ denotes the number of times that action a has been selected prior to time t
- the number $c > 0$ controls the degree of exploration
- If $N_t(a)$=0, then a is considered to be a maximizing action

Upper Confidence Bound

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

- Idea of this upper confidence bound (UCB) action selection is that the square-root term is a measure of the uncertainty or variance in the estimate of a's value
- Quantity being maxed over is thus a sort of upper bound on the possible true value of action a, with c determining the confidence level
- Each time a is selected the uncertainty is presumably reduced: $N_t(a)$ increments, and, as it appears in the denominator, the uncertainty term decreases
- On the other hand, each time an action other than a is selected, t increases but $N_{t(a)}$ does not; because t appears in the numerator, the uncertainty estimate increases
- Use of the natural logarithm means that the increases get smaller over time, but are unbounded
- All actions will eventually be selected, but
  - actions with lower value estimates, or that have already been selected frequently, will be selected with decreasing frequency over time

Results of UCB on 10-arm test bed

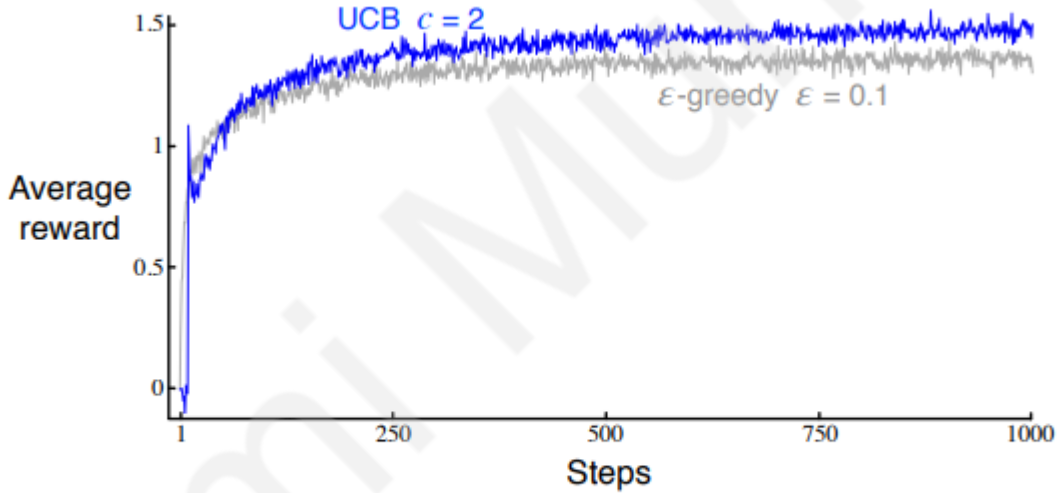


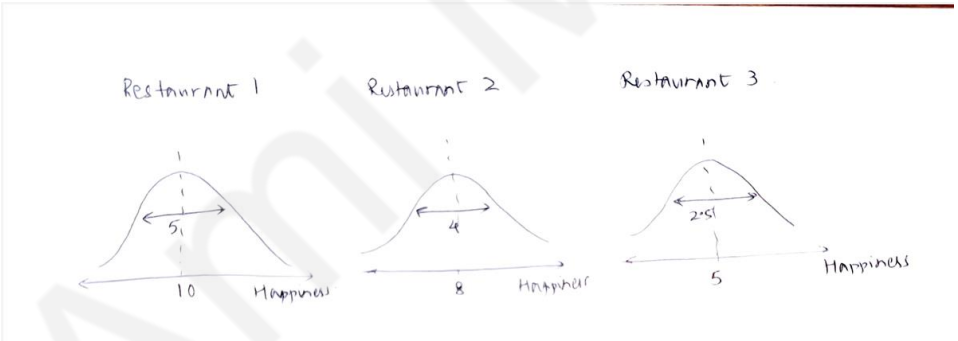
Figure 2.4: Average performance of UCB action selection on the 10-armed testbed. As shown, UCB generally performs better than ε-greedy action selection, except in the first k steps, when it selects randomly among the as-yet-untried actions.

Example on UCB

- Let there be two machines with the following Q values after a total of1000 steps. The number of times each machine is played out of 1000 is also given. Calculate the estimated Q value of each machine using UCB
  - $Q(M_1) = 1, Q(M_2) = 1, Q(M_3) = 1$
  - $N_a(M_1) = 250, N_a(M_2) = 350, N_a(M_3) = 400$
- Solve this example and put the solution on teams

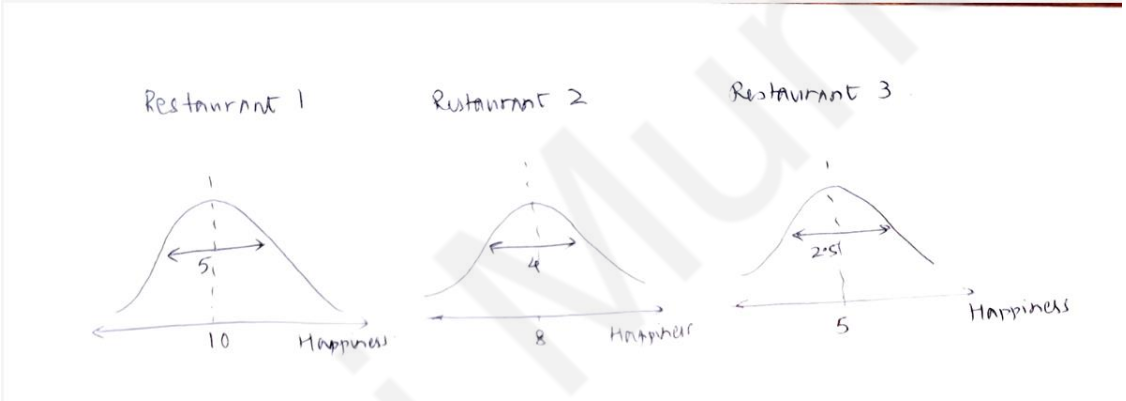
Example

- You are in a new small town for 300 days. You have a choice to eat at one of the three restaurants during those 300 days. Following is the happiness probability distribution of these three restaurants. However this data is not available to you. We will calculate how much happiness is attained using various approaches



Ref: https://www.youtube.com/watch?v=e3L4VocZnnQ

Example



- Maximum happiness that can be attained is $= 300 * 10 = 3000$

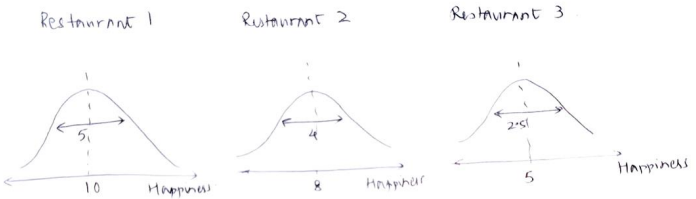
Ref: https://www.youtube.com/watch?v=e3L4VocZnnQ

Explore only

- Exploration??
  - Go to all the restaurants one by one and explore them
- Go to all the three restaurants at random
- So let us assume that you went to all the three restaurants 100 times each
- So the total happiness attained is
  - $Happiness\ attained\ through\ exploration$

$= 100 * 10 + 100 * 8 + 100 * 5 = 2300$

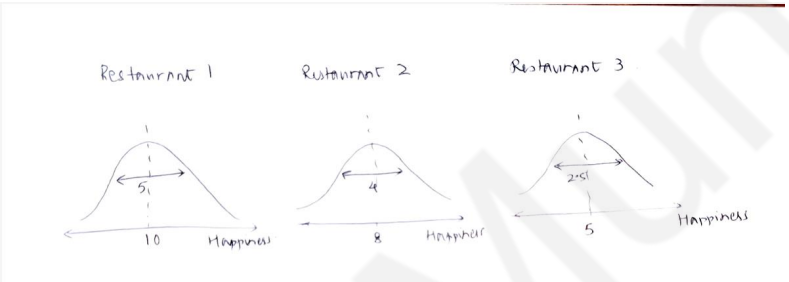
- Regret $\rho = 3000 - 2300 = 700$



Exploit only

- Exploitation
  - Once you have a good idea of which restaurant suits you the best, keep going to the same restaurant again and again
- Assume that you decide to go to each restaurant once and then decide based on you perception where would you go hence forth
- Assume that you get the following happiness from the three restaurants in the first three days
  - Happiness from restaurant 1= 7
  - Happiness from restaurant 2= 8
  - Happiness from restaurant 3= 5
- Since you perception of happiness is highest in restaurant 2, for the remaining 297 days you decide to go to restaurant 2
- Then the total happiness attained through exploit only is
- $Happiness\ through\ exploitation = (7 + 8 + 5) + 297 * 8 = 2396$
- Regret $\rho = 3000 - 2396 = 604$

∈ – Greedy Action



- Take some ∈= 10%
- Explore for ∈ % $times = 30\ times$
- Exploit for the remaining time= 270 times
- Since you have explored 30 times, you know that on an average restaurant 1 gives happiness of 10
- So the remaining 270 days you go to restaurant 1
- $Happiness\ attained\ through\ \in greedy = (10 * 10 + 10 * 8 + 10 * 5) + 270 * 10 = 2960$
- Regret $\rho = 3000 - 2960$

Example on Exploration

Example-Exploration

- Given
  - Time step t=5
  - Number of bandits (machines)=4
  - Probability of selecting each machine is
    - $P_s(M_1) = 0.3, P_s(M_2) = 0.4, P_s(M_3) = 0.2, P_s(M_4) = 0.1$
  - Let Probability of winning in each machine be –
    - $P_w(M_1) = 0.6, P_w(M_2) = 0.4, P_w(M_3) = 0.7, P_w(M_4) = 0.9$
- Assume the random number generated for selection of machines for five iterations is [0.68, 0.55, 0.23, 0.95, 0.8]
- Obtain the estimate of reward generated if following random numbers were generated for the selected machine for each of the five iterations is [0.6, 0.2, 0.3, 0.4, 0.9]

Solution

- Construct a CDF table based on the following machine selection probability
  - $P_s(M_1) = 0.3, P_s(M_2) = 0.4, P_s(M_3) = 0.2, P_S(M_4) = 0.1$

Machine	PDF	CDF
M1	0.3	0.3
M2	0.4	0.7
M3	0.2	0.9
M4	0.1	1.0

- Random number generated for selection of machines for five iterations is [0.68, 0.55, 0.23, 0.95, 0.8]
- Hence from CDF column we infer that Machines $M_2, M_2, M_1, M_4 and\ M_3$ are selected for iterations 1,2,3,4,and 5 respectively

Solution continued

- Now based on the machines selected and the random number generated for each iteration, we will decide whether the machine fetched a reward of 1 or 0
- Probability of winning in each machine is $P_w(M_1) = 0.6, P_w(M_2) = 0.4, P_w(M_3) = 0.7, P_w(M_4) = 0.9$
- Random number generated were as follows
  - [0.6, 0.2, 0.3, 0.4, 0.9]

Iteration	Random number generated for machine selection	Machine selected	Probability of winning of each machine P_w	Random number generated for reward P_r	Reward obtained If $P_r \leq P_w$, Reward=1
1	0.68	M_2	0.4	0.6	0
2	0.55	M_2	0.4	0.2	1
3	0.23	M_1	0.6	0.3	1
4	0.95	M_4	0.9	0.4	1
5	0.8	M_3	0.7	0.9	0

Solution continued

- Total reward obtained = 3
- Maximum reward that was possible if the best machin e is selected(M_4 with probability of winning $P_w(M_4) = 0.9$) is
  - 0.9*5= 4.5
- Regret= 4.5-3=1.5

Note

- When we compute the probability of getting a reward, we are comparing the probability against a random number
- If probability of winning is greater than the random number then we assume we will get the reward
- Whereas when you compute the regret, we have taken the maximum possible reward as 0.9 x 5 = 4.5, which is statistically right
- These bandits, which either give a reward or not, are also called Bernoulli Bandits

Example on Exploitation

Example-Exploitation

- Given
  - Time step t=5
  - Number of bandits (machines)=4
  - Probability of selecting each machine is
    - $P_s(M_1) = 0.3, P_s(M_2) = 0.4, P_s(M_3) = 0.2, P_S(M_4) = 0.1$
  - Let Probability of winning in each machine be –
    - $P_w(M_1) = 0.6, P_w(M_2) = 0.4, P_w(M_3) = 0.7, P_w(M_4) = 0.9$
- Assume the random number generated for selection of one machine out of five in the first iteration is 0.65
- Once the machine is selected in the first iteration, the same machine will be exploited for the remaining 2^{nd}, 3^{rd}, 4^{th} and 5^{th} iterations
- Obtain the estimate of reward generated if following random numbers were generated for the selected machine for all five iterations is [0.6, 0.2, 0.3, 0.5, 0.9]

Solution

- Construct a CDF table based on the following machine selection probability
  - $P_s(M_1) = 0.3, P_s(M_2) = 0.4, P_s(M_3) = 0.2, P_S(M_4) = 0.1$

Machine	PDF	CDF
M1	0.3	0.3
M2	0.4	0.7
M3	0.2	0.9
M4	0.1	1.0

- Random number generated for selection of one machine out of five in the first iteration is 0.65
- Hence from CDF column we infer that Machine M_2, is selected for all the iterations 1,2,3,4,and 5

Solution continued

- Now based on the machines selected and the random number generated for selection of machine, Machine M_2, is selected for all the iterations 1,2,3,4,and 5
- Probability of winning in each machine is for $\mathrm{M}_2$is $P_w(M_2) = 0.4$
- Random number generated for the reward were as follows
  - [0.6, 0.2, 0.3, 0.5, 0.9]

Iteration	Random number generated for reward P_r	Reward obtained If $P_r \leq P_w$ That is P_r <=0.4 Reward=1
1	0.6	0
2	0.2	1
3	0.3	1
4	0.5	0
5	0.9	0

Solution continued

- Total reward obtained = 2
- Maximum reward that was possible if the best machine is selected (M_4 with probability of winning $P_w(M_4) = 0.9$) is
  - 0.9*5= 4.5
- Regret= 4.5-2=2.5

Fixed Exploration +Exploitation

Example-Fixed Exploration +Exploitation

- Given
  - Time step t=10
  - Number of bandits (machines)=4
  - Let number of fixed exploration =4
  - Let Probability of winning in each machine be –
    - $P_w(M_1) = 0.6, P_w(M_2) = 0.4, P_w(M_3) = 0.7, P_w(M_4) = 0.9$
  - We assume here that in 4 chances of exploration,
    - Machine M_3 got played 3 times out of which it won twice
    - Machine M_1 got played one time and it won
- Calculate the final reward obtained if the machine which is perceived the best after four chances of exploration is exploited and the random number generated to estimate the rewards for remaining six iterations are [0.33, 0.25, 0.55,0.45, 0.6, 0.8]

Solution

- We know that in 4 chances of exploration,
  - Machine M_3 got played 3 times out of which it won twice
  - Machine M_1 got played one time and it won
- Therefore estimated values for each machines are
  - $Q_*(M_1) = 1, Q_*(M_2) = 0, Q_*(M_3) = \frac{2}{3} = 0.67, Q_*(M_4) = 0$
- From the Q_* estimates , machine M_1 is exploited for the next 6 steps
- Probability of winning for M_1 is $P_w(M_1) = 0.6$
- Random number generated to estimate the rewards for remaining six iterations are [0.33, 0.25, 0.55,0.45, 0.6, 0.8]

Iteration	Random number generated for reward P_r	Reward obtained If $P_r \leq P_w$ That is P_r <=0.6 Reward=1
1	0.33	1
2	0.25	1
3	0.55	1
4	0.45	1
5	0.6	1
6	0.8	0

Solution continued

- Total reward from all the 10 steps is
  - 3 (from exploration)+5 (From exploitation) =8
- If the best machine were played then the reward obtained would have been
  - 10*0.9=9
- Therefore regret is
  - 9-8=1