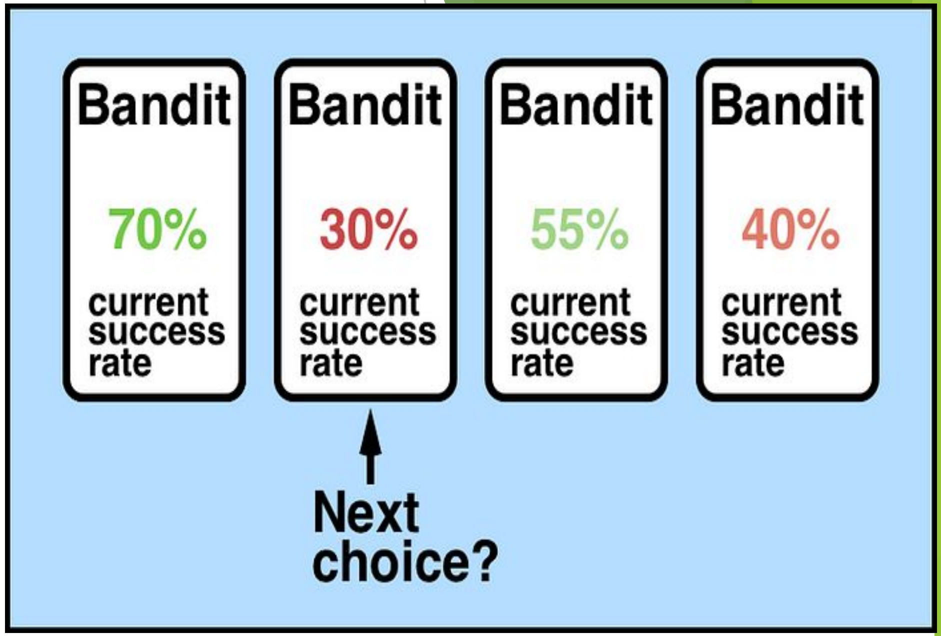
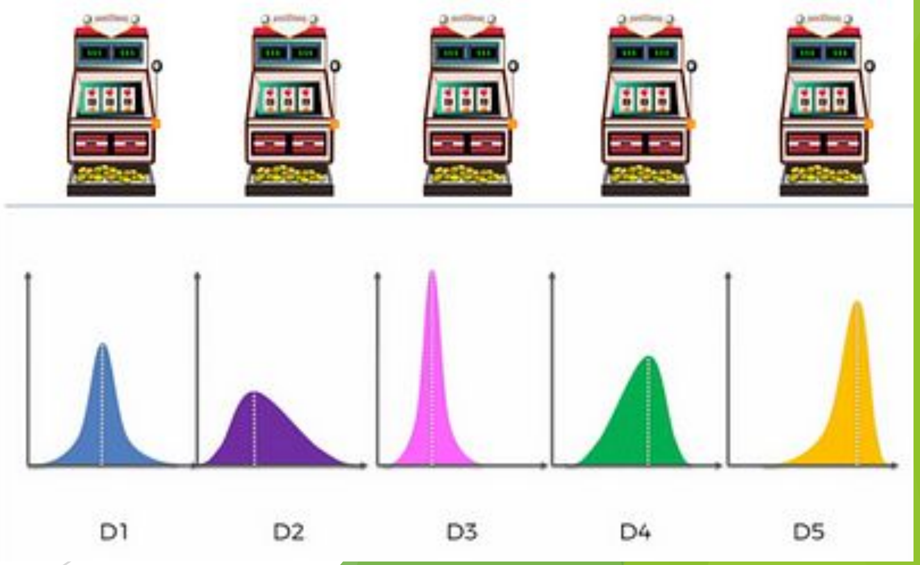
Multi-Arm Bandits

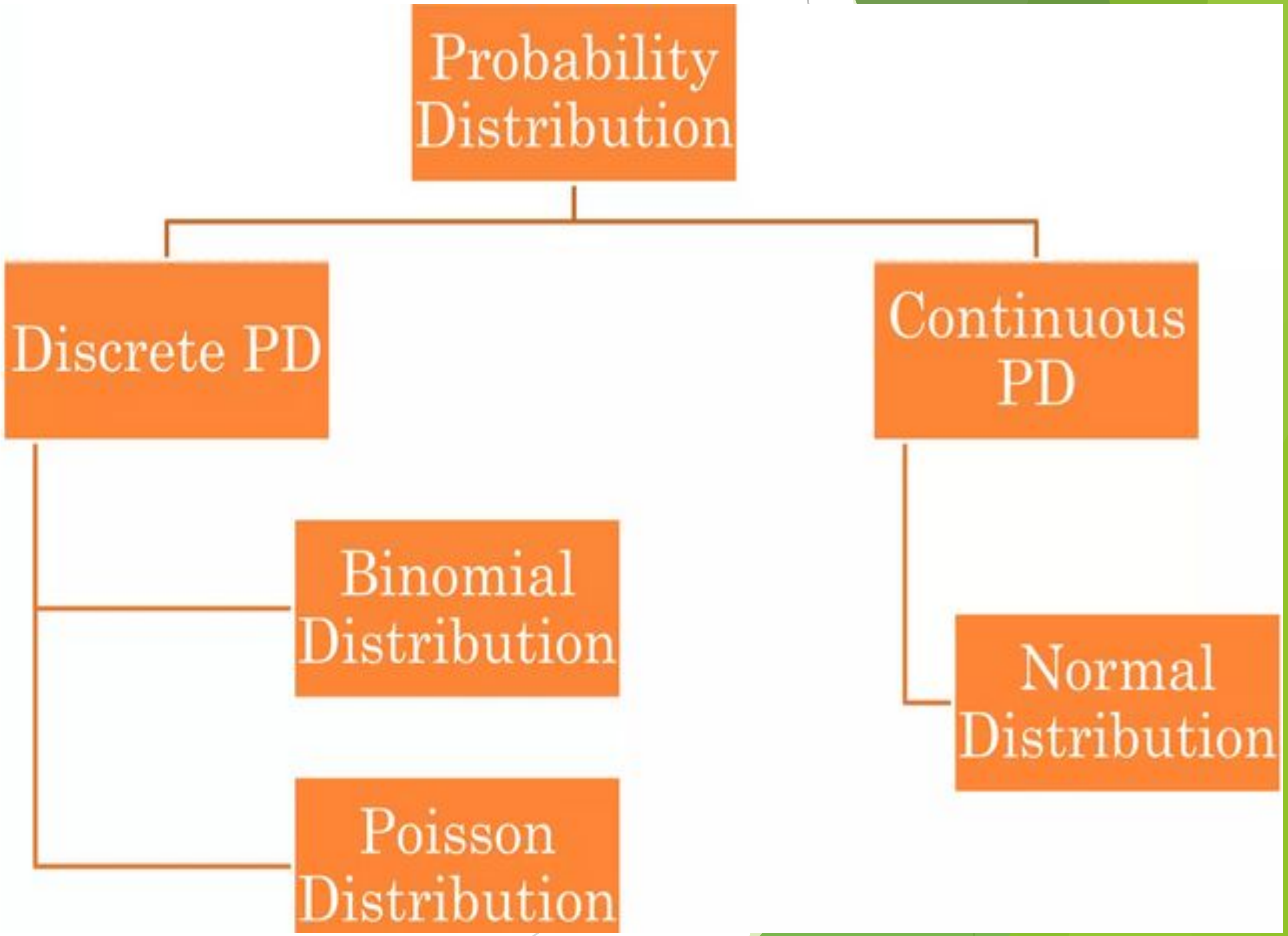
- Reinforcement Learning(RL) uses training information that 'Evaluate' the actions
- Evaluative feedback indicates 'How good the action taken was'
- The multi-armed bandit problem is a classic reinforcement learning example
- The multi-armed bandit, a slot machine with *n* arms (bandits) with each arm having its own probability distribution of success
- Pulling any one of the arms gives a stochastic reward of either R=+1 for success, or R=0 for failure





Probability Distribution

- It is a listing of the probabilities of all the possible outcomes that could occur if the experiment was done
  - Discrete_Distribution: Random Variable can take only limited number of values
    - Ex: No. of heads in two tosses
    - No. of dots on Dies
  - Continuous Distribution: Random Variable can take any value
    - Ex: Height of students in the class



Discrete Distribution

- Random Variable can take only limited number of values
  - Ex: No. of heads in two tosses
  - No. of dots on Dies
- Tossing a coin three times:
  S = $\{HHH, HHT, HTH, HTT,$

  $THH, THT, TTH, TTT\}$
- Let X represents "No. of heads"

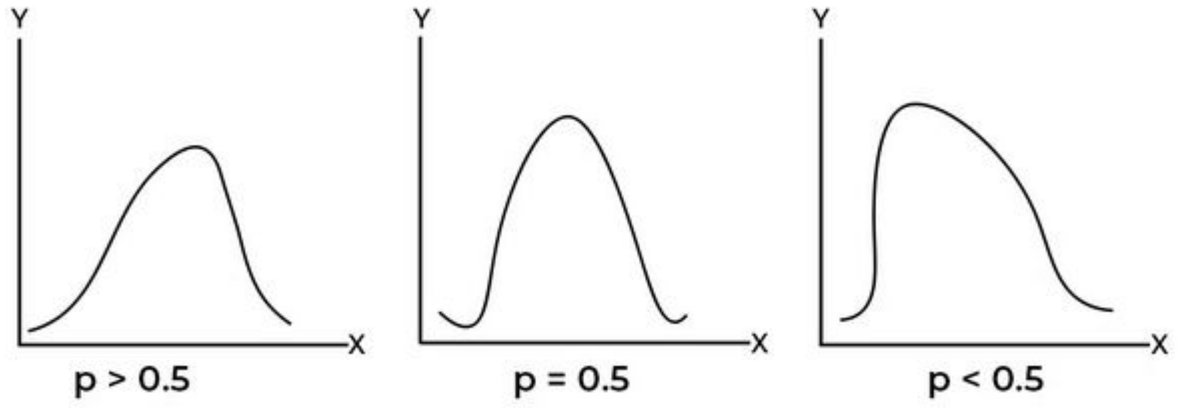
X	Frequency	P (X=x)
0	1	1/8
1	3	3/8
2	3	3/8
3	1	1/8

Binomial Distribution

- There are certain phenomena in nature which can be identified as Bernoulli's processes, in which:
  - There is a fixed number of n trials carried out
  - Each trial has only two possible outcomes say success or failure, true or false etc.
  - Probability of occurrence of any outcome remains same over successive trials
  - Trials are statistically independent
  - Binomial distribution is a discrete PD which expresses the probability of one set of alternatives – success (p) and failure (q)

$$P(X = x) = {}^{n}_{r}C\, p^{r}\, q^{n-r}$$

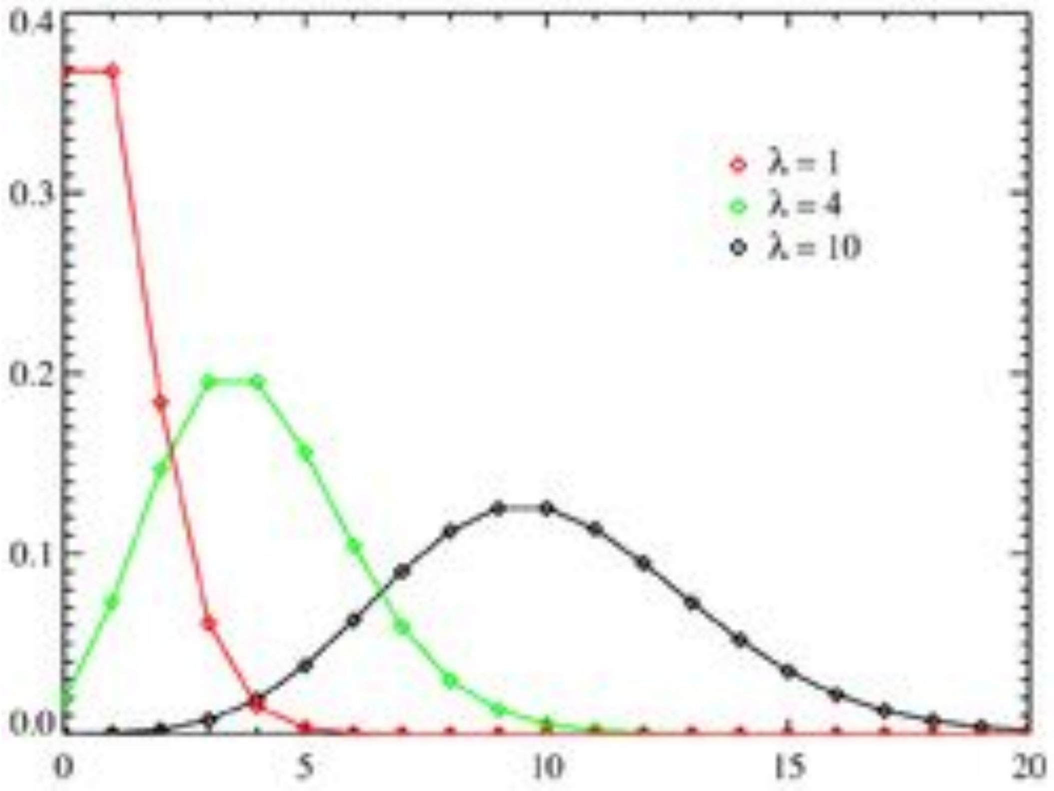
-
  - n = no. of trials undertaken
  - r = no. of successes desired
  - p = probability of success
  - q = probability of failure



Poisson Distribution

- When there is a large number of trials, but a small probability of success, binomial calculation becomes impractical
- If λ = mean no. of occurrences of an event per unit interval of time/space, then probability that it will occur exactly 'x' times is given by

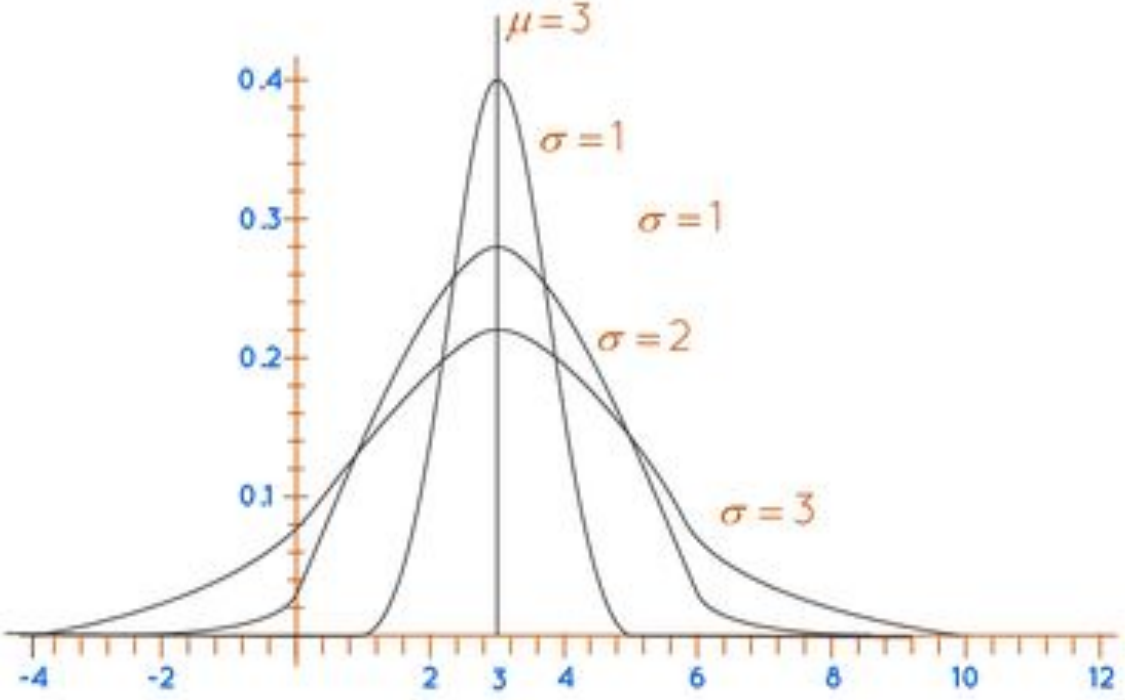
$$P(x) = \frac{\lambda^x e^{-\lambda}}{x!}$$



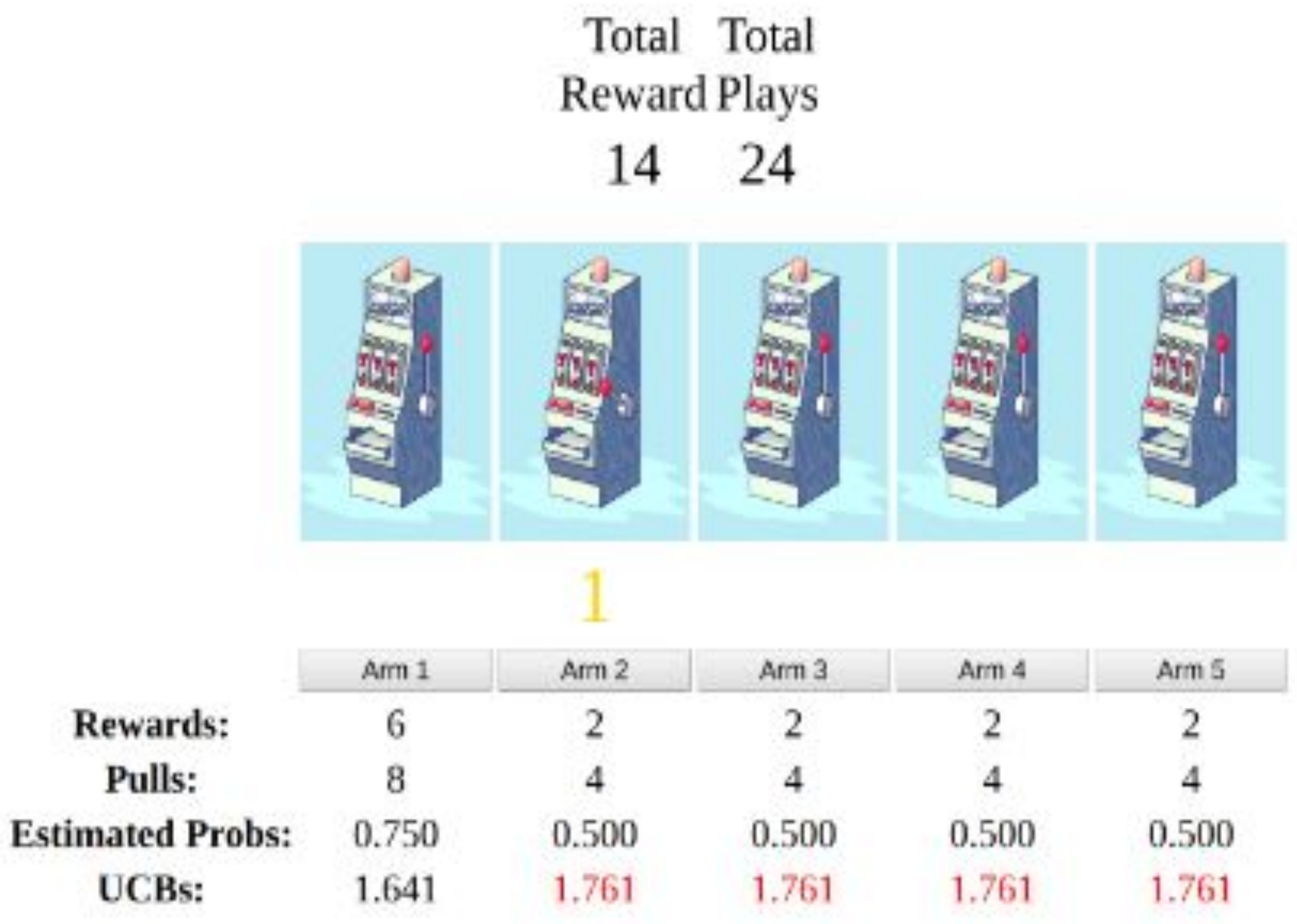
Normal Distribution

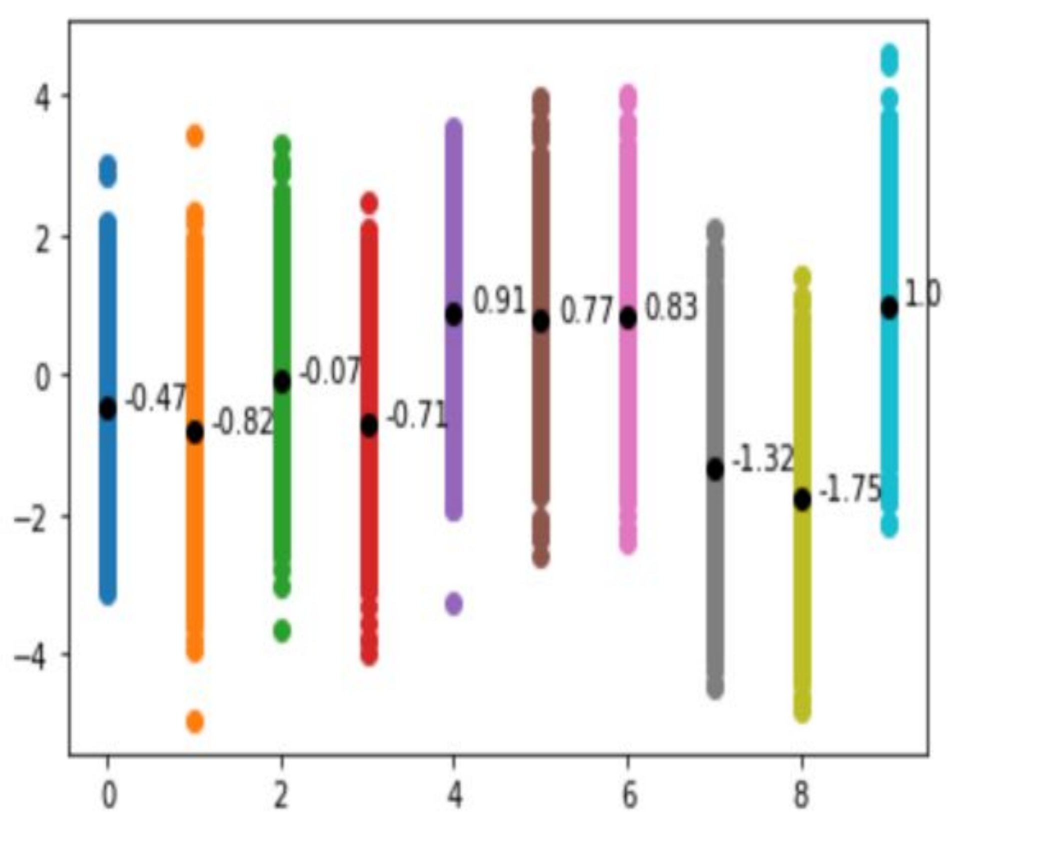
- It is a continuous PD i.e. random variable can take on any value within a given range. Ex: Height, Weight, Marks etc.
- Developed by Karl Gauss, so also called Gaussian Distribution
- It is symmetrical, unimodal (one peak)
- X axis represents random variable like height, weight etc.
- Y axis represents its probability density function
- Mean = μ, SD = σ

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$



MAB Interactive Demo





https://perso.crans.org/besson/phd/MAB_interactive_demo/

Cont...

- MAB deal with ‘Exploitation & Exploration’ of the core ideas in RL
- For example:
  - Among the various slot machines, which slot machine should selected and lower the lever?
  - How can to make the best return?
- In the full reinforcement learning problem, a MAB algorithm is always used to optimizes the solution

Multi Bandit Problem

- Multi bandit problems are problems in the area of sequential selection of experiments
- At each stage there are k possible actions or choices of experiment
- An information or a reward is resulted based on the choice of action
- There must strike a balance between gaining rewards and gaining information to maximize the present value of the rewards received
- That leads to exploitation/exploration dilemma
- MAB deal with 'Exploitation & Exploration' of the core ideas in RL
- For example:
  - Among the various slot machines, which slot machine should selected and lower the lever?
  - How can to make the best return?

The Exploration/Exploitation Dilemma

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

Examples

- Restaurant Selection
  - Exploitation: Go to your 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

K-armed Bandit Problem

- The k-armed bandit, with k arms, each having an unknown, possibly different distribution of payoffs/rewards
- In k-armed bandit problem, each of the k actions has an expected or mean reward given that, based on action selected is the value of that action (Action Value):

$$q_*(a) \doteq \mathbb{E}[R_t \mid A_t = a]$$

- Where,
  - t- step or play number
  - $A\ t$ - Action at time t
  - Rt - Reward at time t
  - $q*$ (a)- True value (expected reward) of action

Action-value Methods

- Action-value methods are simple methods for estimating the values of actions
- This estimates to make action selection decisions
- q(a) denote the true (actual) value of action a, the true value of an action is the mean reward received when that action is selected
- The estimated value on the t^{th} time step as $Q_t(a)$
- If by the t^{th} time step action a has been chosen $N_t(a)$ times prior to t, rewards R_1, R_2, . . . , $RN_t(a)$, then its value is estimated to be

$$Q_t(a) = \frac{R_1 + R_2 + \cdots + R_{N_t(a)}}{N_t(a)}.$$

Action-value Methods

- For example, estimate action values as *sample averages:*

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

- The sample-average estimates converge to the true values *If* the action is taken an infinite number of times

$$\lim_{N_t(a)\to\infty} Q_t(a) = q_*(a)$$

The number of times action a has been taken by time t

Example 2(May_2024) 10M

- You are playing a slot machine with three arms.
- Each time you pull an arm, you either win $1 or lose $1 with equal probability.
- You decide to randomly choose an arm to pull each time.
- If you play the slot machine 100 times, how much money do you expect to win or lose on average?

Example: Clinical Trials

- There are **k** treatments for a given disease
- Patients arrive sequentially at the clinic and must be treated immediately by one of the treatments
- It is assumed that response from treatment is immediate
- So the effectiveness of the treatment of the present patient receives is known when the next patient to be treated
- It is not known precisely which one of the treatments is best, but one must decide which treatment to give each patient
- The goal is to cure as many patients as possible
- This may require to give a patient a treatment which is not the one that looks best at the present time in order to gain information that may be of use to future patients

Types of Action-value Methods

- Greedy Action Selection Method
- ε-greedy Action Selection Method
- Upper-Confidence-Bound(UCB) Action Selection Method

Greedy Action Selection

- The simplest action selection rule is to select the action with highest estimated action value is, called the greedy actions selection
- At step t one of the greedy actions, A_t^*, for which
- $Q_t(A_t^*) = max_a\, Q_t(a)$

$$A_t \doteq \underset{a}{\operatorname{argmax}}\, Q_t(a)$$

$argmax_a$ f(a) - a value of a at which $f(a)$ takes its maximal value

- 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

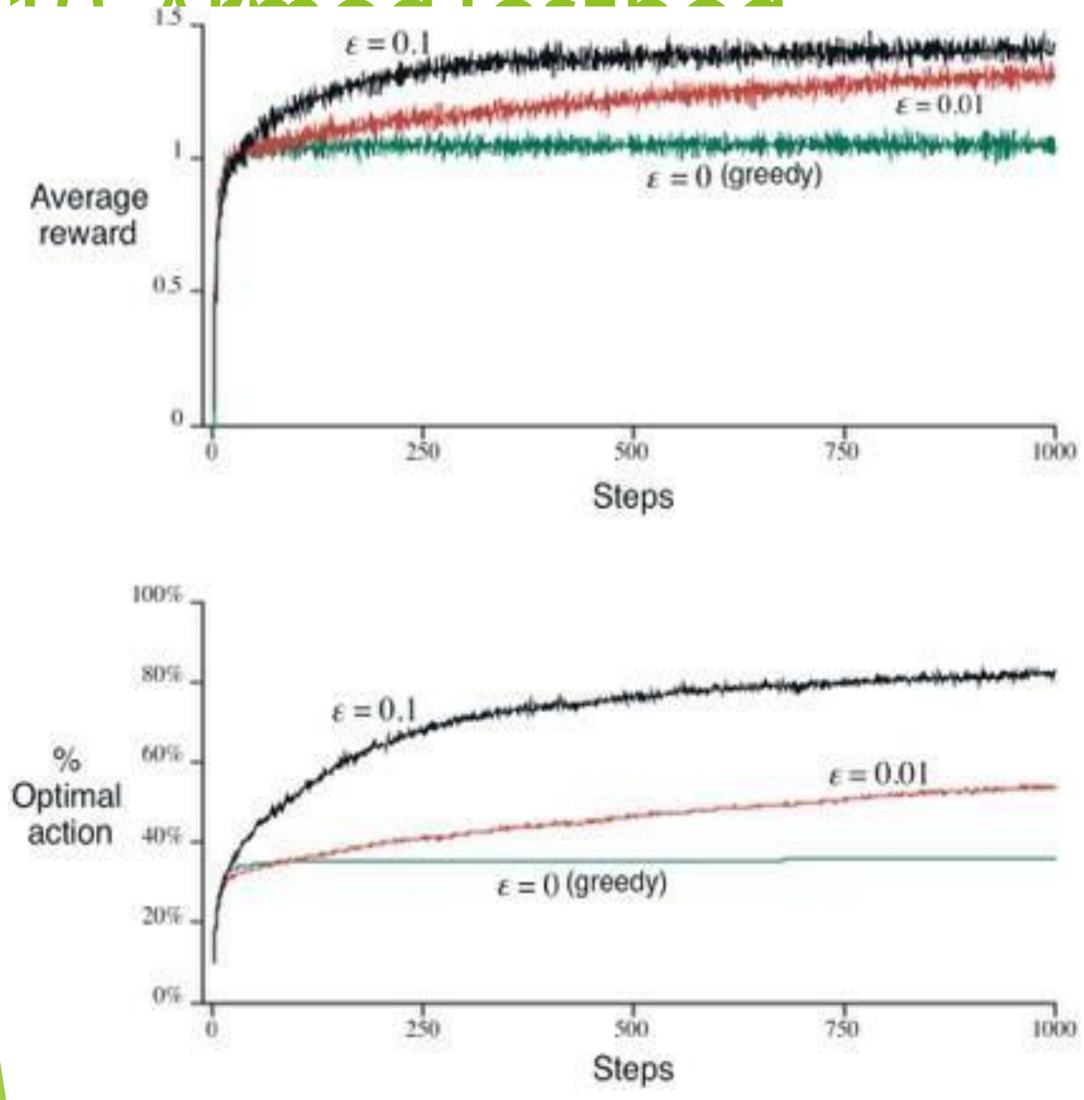
ε-greedy Action Selection Method

- 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
- A simple alternative method, ε-greedy action selection method behave greedily most of the time
- However, with small probability ε, select the action randomly from all the actions with equal probability, independently of the action value estimates

$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \longrightarrow \text{Exploitation} \\ \text{a random action} & \text{with probability } \varepsilon \longrightarrow \text{Exploration} \end{cases}$$

- An advantage of these methods is that, as the number of plays increases, every action will be sampled an infinite number of times, Nt(a) $\rightarrow \infty$ for all a, ensure the Q_t(a) converge to q(a)

ε-Greedy Methods on the 10-ArmedTestbed



Conceder a set of 2000 randomly generated n-armed bandit tasks with n = 10

For each bandit, the action values, q(a), a = 1, . . . , 10, were selected according to a normal (Gaussian) distribution with mean 0 and variance 1

Figure shows the performance and behavior of various methods as they improve with experience over 1000

Steps

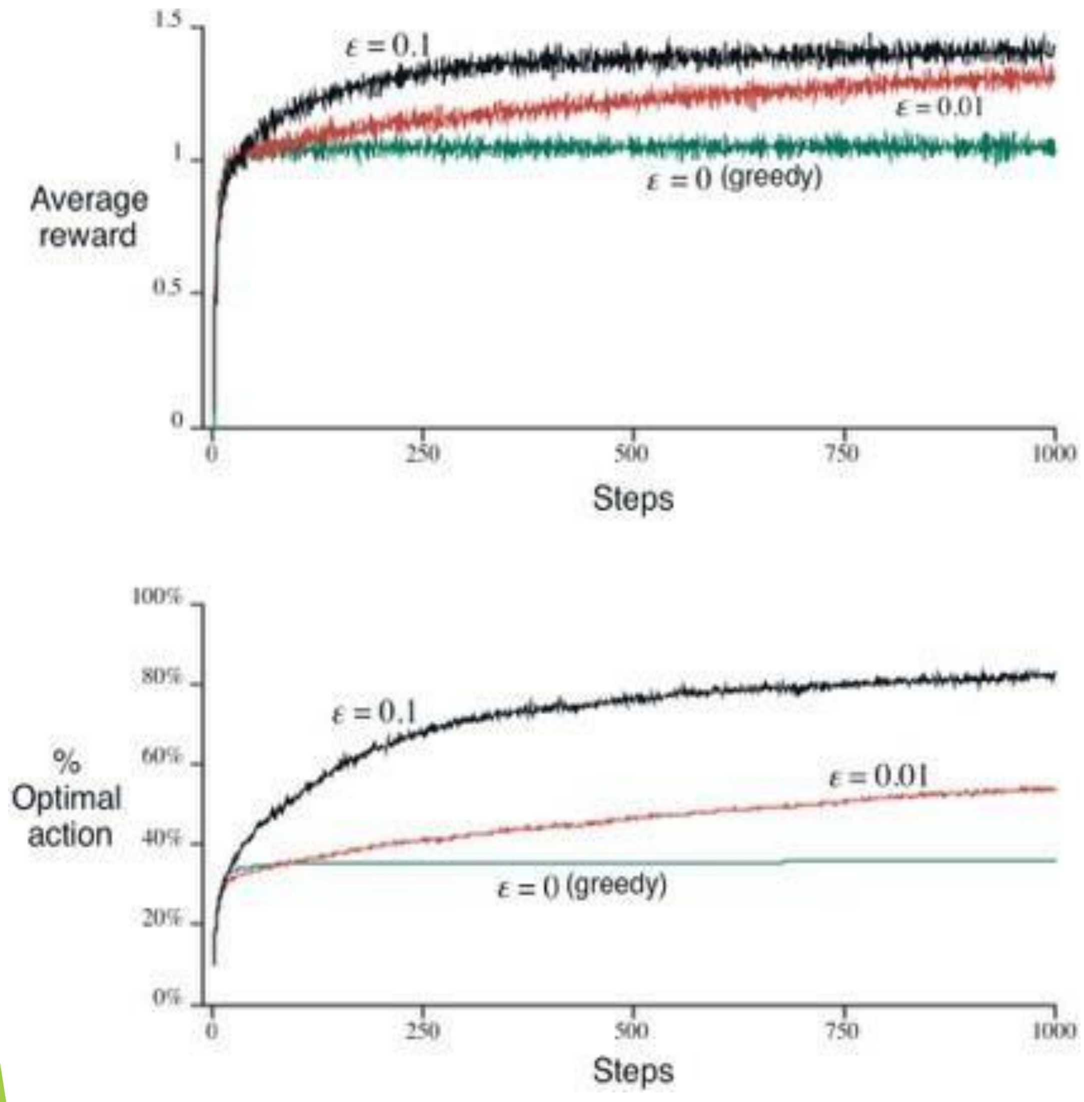
Figure compares a greedy method with two ε-greedy methods (ε = 0.01 and ε = 0.1)

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 1.55 on this testbed

ε-Greedy Methods on the 10-ArmedTestbed



The greedy method performs significantly worse in the long run because it often gets stuck performing suboptimal actions

The ε-greedy methods perform better because they continue to explore, and to improve their chances of recognizing the optimal action

The ε = 0.1 method explores more and finds the optimal action earlier, but never selects it more than 91% of the time

The ε = 0.01 method improves more slowly, but performs better than the ε = 0.1 method on both performance measures

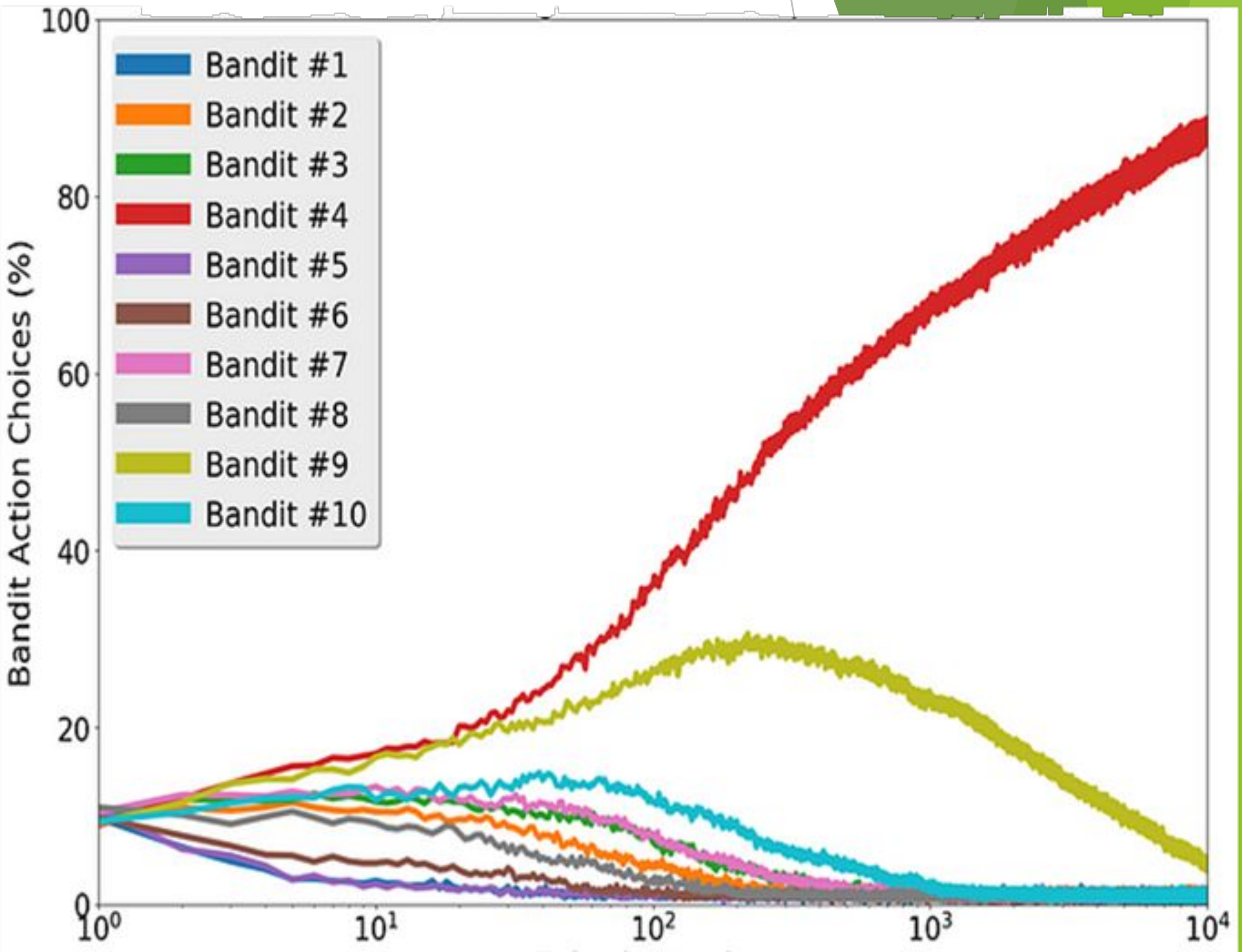
It is also possible to reduce ε over time to try to get the best of both high and low values

Example:

Consider an example of 10 bandits each with their own individual success probabilities and ε = 0.1

Bandit #1 = 10% success rate
Bandit #2 = 50% success rate
Bandit #3 = 60% success rate
Bandit #4 = 80% success rate
Bandit #5 = 10% success rate
Bandit #6 = 25% success rate
Bandit #7 = 60% success rate
Bandit #8 = 45% success rate
Bandit #9 = 75% success rate
Bandit #10 = 65% success rate

By training the agent for 10,000 episodes per experiment, the average proportion of bandits chosen by the agent as a function of episode number, is shown in Fig



Example:3

- An agent playing a slot machine with three arms each with their own individual success probabilities and ε = 0.4. Bandit #1 = 40% success rate, Bandit #2 = 70% success rate and Bandit #3 = 80% success rate. Each time agent pull an arm, it either reward $1 or -$1. If it play the slot machine 10000 times, calculate expected cumulative reward.

Comparison Between Greedy and ε-Greedy Methods

- The advantage of ε-greedy over greedy methods depends on the task
- If the reward variance is been larger, 10 instead of 1, it takes more exploration to find the optimal action, and ε-greedy methods is better relative to the greedy method
- If the reward variances is zero, then the greedy method would know the true value of each action after trying it once
- In this case the greedy method perform best because it would soon find the optimal action and then never explore

The Exploration/Exploitation Dilemma

- Suppose you form estimates

$$Q_t(a) \approx q_*(a), \quad \forall a$$

action-value estimates

- Define the *greedy action* at time *t* as

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

- If $A_t = A_t^*$ then you are *exploiting*
  If $A_t \neq A_t^*$ then you are *exploring*

- You can't do both, but you need to do both

- You can never stop exploring, but maybe you should explore less with time. Or maybe not.

Example:4 (IT_23_10M)

- 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

- The action-value methods estimate action values as sample averages of observed rewards

$$Q_t(a) = \frac{R_1 + R_2 + \cdots + R_{N_t(a)}}{N_t(a)}$$

- For each action ‘a’, a record of all the rewards that have followed the selection of that action needs to be maintained
- Each additional reward following a selection of action ‘a’ requires more memory to store it and results in more computation required to determine $Q_t(a)$
- Instead of recalculating the sum every time, incrementally update the sample average

Cont…

- For some action, let Q_k denote the estimate for its K^{th} reward, the average of its first k – 1 rewards
- Given this average and a K^{th} reward for the action, R_k, then the average of all 'k' rewards can be computed by
- This implementation requires memory only for Q_k & 'k', and only the small computation for each new reward

$$Q_t(a) = \frac{R_1 + R_2 + \cdots + R_{N_t(a)}}{N_t(a)}$$

$$\begin{aligned} Q_{k+1} &= \frac{1}{k}\sum_{i=1}^{k} R_i \\ &= \frac{1}{k}\left(R_k + \sum_{i=1}^{k-1} R_i\right) \\ &= \frac{1}{k}\left(R_k + (k-1)Q_k + Q_k - Q_k\right) \\ &= \frac{1}{k}\left(R_k + kQ_k - Q_k\right) \\ &= Q_k + \frac{1}{k}\left[R_k - Q_k\right], \end{aligned}$$

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

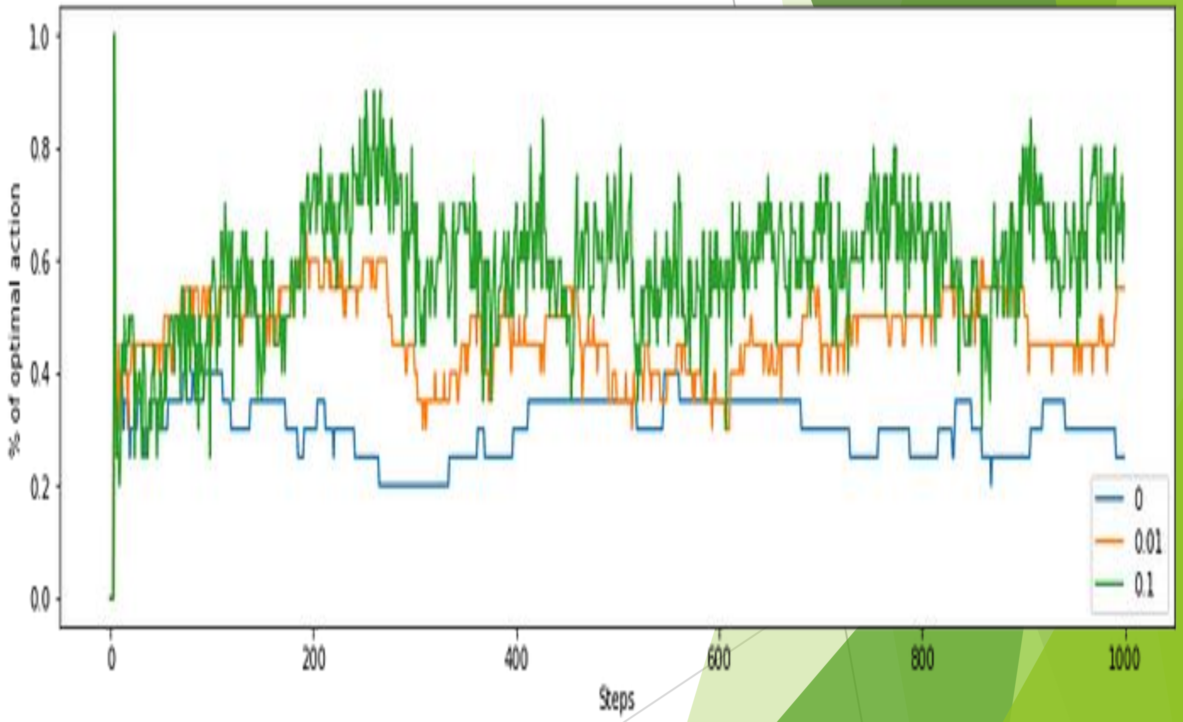
Expression [Target – OldEstimate] is an error in the estimate. The step-size parameter (StepSize) changes from time step to time step

Stationary vs Nonstationary Problem

- In a stochastic processes, a stationary and a nonstationary problem refer to the nature of the statistical properties of a process over time
- Stationary Problem:
  - A stationary Problem has statistical properties (like mean, variance, etc) that do not change over time
  - Example: Daily temperature deviation from the monthly mean
- Nonstationary Problem:
  - In a nonstationary problem, the environment's underlying characteristics (statistical properties) change over time
  - Example: Stock Market Index Prices, GDP growth

Tracking a Nonstationary Problem

- In a stationary environment, the reward distributions for the multi bandits arms are constant, so an agent can learn over time and use its knowledge to make better decisions
- But in a nonstationary environment, the reward distributions of the arms are not fixed but changed unpredictably
- So the optimal action can change and the agent needs to adjust its behavior accordingly to track the best-performing options



Cont…

- The averaging methods discussed are appropriate in a stationary environment, but not if the bandit is changing over time
- As most of the reinforcement learning problems are nonstationary, weight recent rewards more heavily than long-past ones by using a constant step-size parameter

$$Q_{k+1} = Q_k + \alpha\Big[R_k - Q_k\Big]$$

- where the step-size parameter $\alpha \in (0, 1]$ is constant

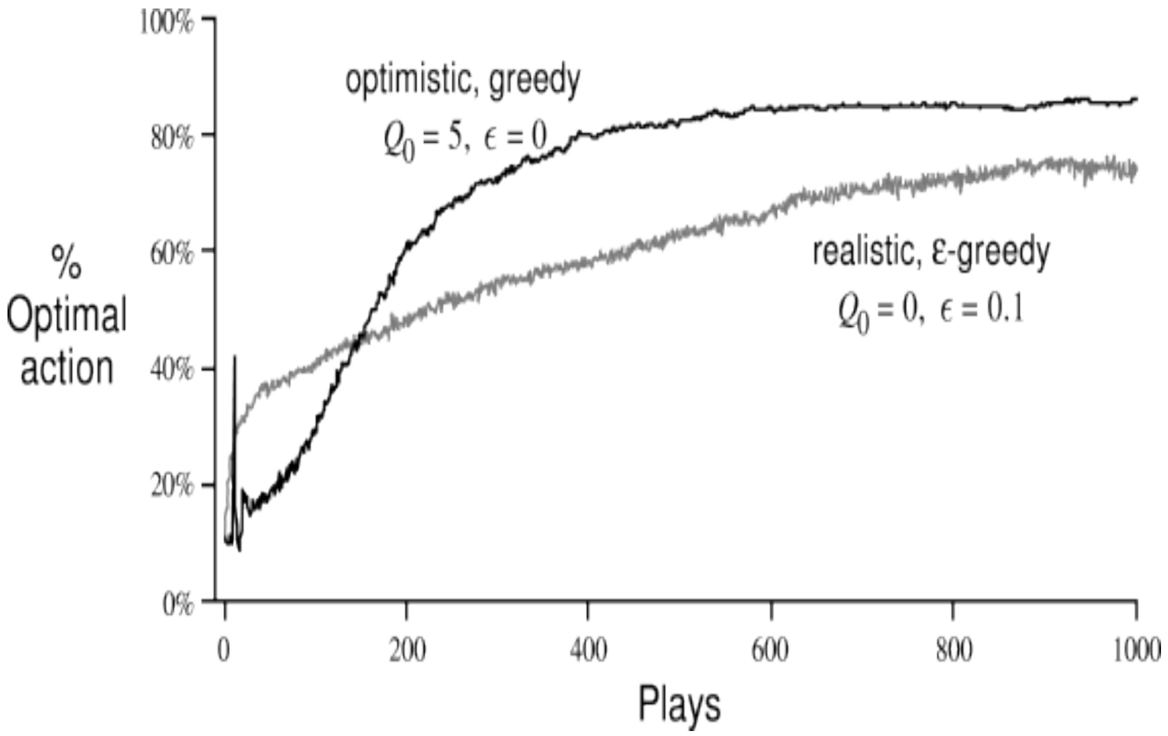
$$
\begin{aligned}
Q_{k+1} &= Q_k + \alpha\Big[R_k - Q_k\Big] \\
&= \alpha R_k + (1-\alpha)Q_k \\
&= \alpha R_k + (1-\alpha)\left[\alpha R_{k-1} + (1-\alpha)Q_{k-1}\right] \\
&= \alpha R_k + (1-\alpha)\alpha R_{k-1} + (1-\alpha)^2 Q_{k-1} \\
&= \alpha R_k + (1-\alpha)\alpha R_{k-1} + (1-\alpha)^2 \alpha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

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

Optimistic Initial Values

- Epsilon-Greedy Algorithms are dependent on the initial action-value estimates Q_0 and are biased by their initial estimates
- Optimistic Initial Values are used to encourage exploration in the early stages of learning
- In many RL algorithms, the agent learns by balancing exploration and exploitation
- Without exploration, an agent might get stuck in suboptimal policies and never discover better alternatives
- By assigning optimistic initial values to actions or states, the agent is initially “overestimating” the reward it will get from those actions
- This makes the agent more likely to explore less-visited or uncertain actions in the early stages of training

Cont…



- The agent is more "optimistic" about what it might find, leading it to explore different actions before settling on an optimal policy

- Figure shows the performance on the 10-armed bandit testbed of a greedy method using Q_0 = +5, for all α and the ε-greedy method with Q_0 =0
- Both methods used a constant step-size parameter, α
- Initially, the optimistic method performs worse because it explores more, but eventually it performs better because its exploration decreases with time
- This technique for encouraging exploration effective on stationary problems, but it is not well suited to nonstationary problems

Upper Confidence Bound (UCB) action selection

- Simple strategies like ε-greedy choose either to explore or exploit based on a fixed probability, ε = 0.4
- This strategy can lead to issues:
  - Over-exploration: It might waste too many pulls on less promising arms.
  - Under-exploration: If the exploration rate is too low, the agent might not discover better arms that it has not pulled enough to evaluate properly.
- The Upper Confidence Bound (UCB) algorithm address the shortcomings of ε-greedy methods
- It uses uncertainty in the action-value estimates for balancing exploration and exploitation

UCB's Approach

- UCB dynamically adjusts between exploration and exploitation based on how many times each arm has been pulled and the uncertainty around its performance
- It is guided by the principle that arms that have been pulled fewer times should be explored more
- But as the agent gains more data or 't' approaches larger value, it can exploit the arms that consistently perform well
- In the UCB, the square-root term is a measure of the uncertainty or variance in the estimate of a's value

$$A_t = \text{argmax}_a \left(\underbrace{Qt(a)}_{\text{Exploit}} + \underbrace{c\sqrt{\frac{\ln(t)}{N_t(a)}}}_{\text{Explore}}\right)$$

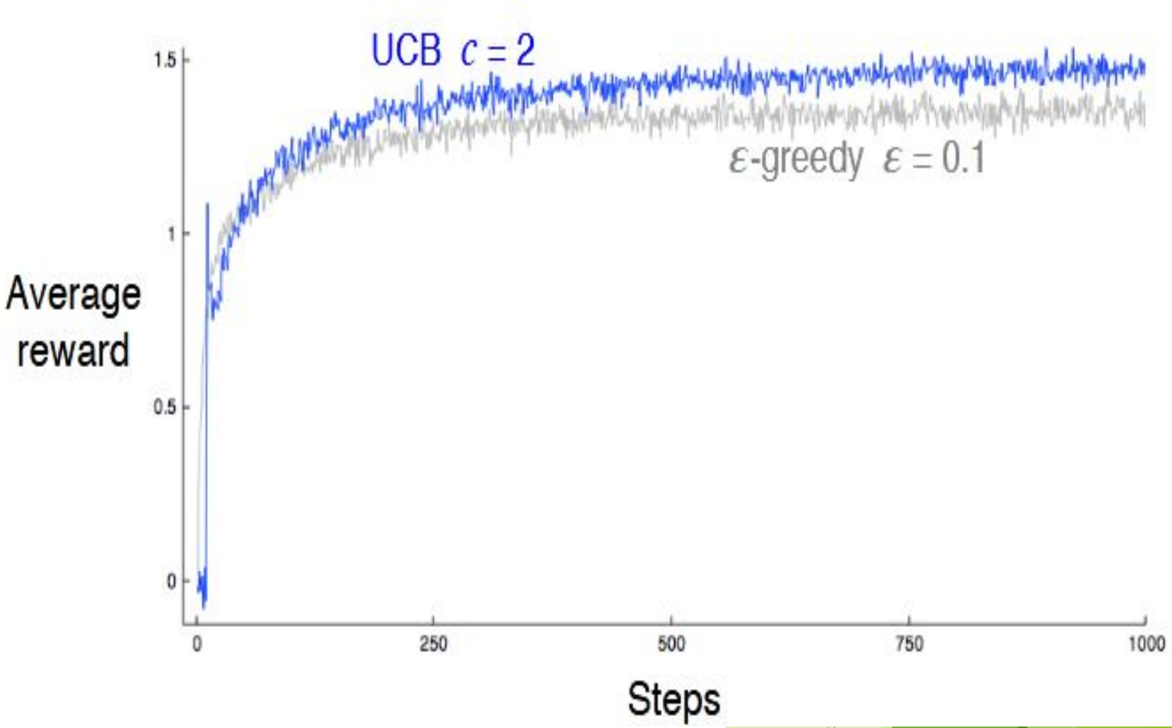
c > 0 controls the degree of exploration

The probability that the slot machine may be the optimal slot machine

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

Cont…

- Each time a is selected the uncertainty is reduced as Nt(a) increases
- But each time an action a is selected ‘t’ is increased; the uncertainty estimate is increased
- The use of the natural logarithm means that the increase gets smaller over time
- As time goes by it will be a longer wait, and thus a lower selection frequency
- So UCB explores more to systematically reduce uncertainty but its exploration reduces over time



Example:4

- After 12 iterations of the UCB 1 algorithm applied on a 4-arm bandit problem, we have n1 = 3, n2 = 4, n3 = 3, n4 = 2 and Q12(1) = 0.55, Q12(2) = 0.63, Q12(3) = 0.61, Q12(4) = 0.40. Which arm should be played next?

Gradient Bandits

- Unlike value-based methods, which learn a value function for states or actions, Gradient Bandits directly optimize the parameters, a numerical preference $H_t(a)$ for each action 'a'
- It's particularly useful in environments where the agent needs to learn a probabilistic policy rather than a deterministic one
- In Gradient Bandits, the policy is represented as a probabilistic model, $\pi_t(a)$ for the probability of taking action 'a' at time 't', according to a soft-max distribution as:

$$\Pr\{A_t = a\} = \frac{e^{H_t(a))}}{\sum_{b=1}^{n} e^{H_t(b)}} = \pi_t(a)$$

Cont…

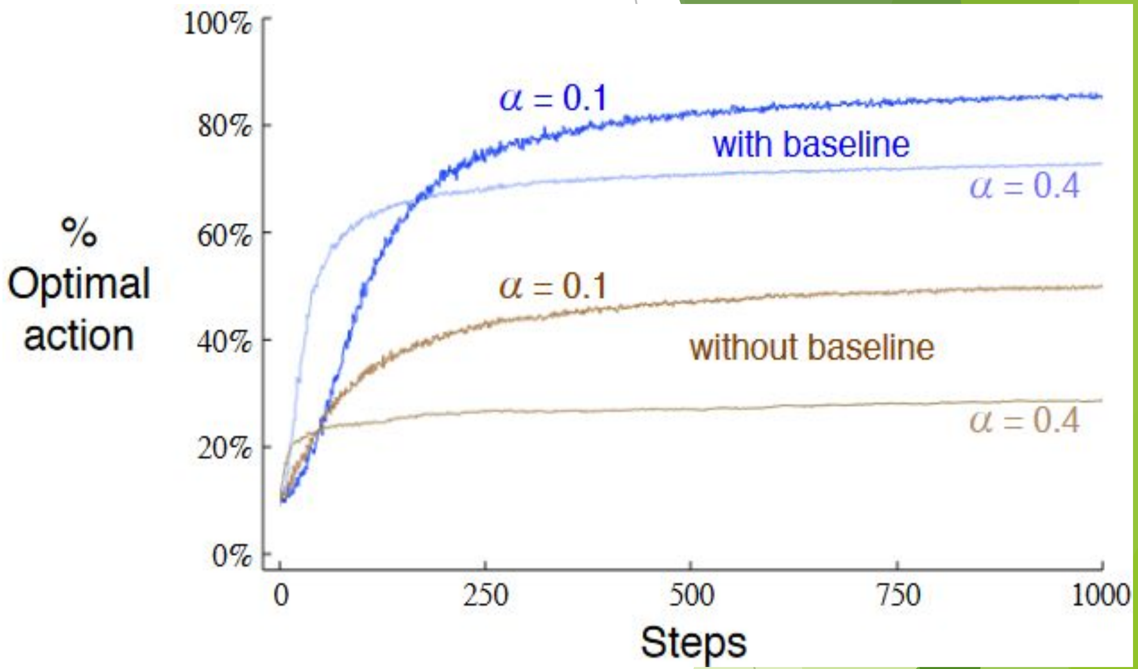
- Gradient Bandits uses the gradient ascent technique to update the parameters $H_t(a)$

$$H_{t+1}(A_t) = H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), \qquad \text{and}$$
$$H_{t+1}(a) = H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), \qquad \forall a \neq A_t$$

- $\bar{R}_t$ is the average of all the rewards up through and including time ‘t’
- The $\bar{R}_t$ term serves as a baseline with which the reward is compared
- If the reward is higher than the baseline, then the probability of taking At in the future is increased
- If the reward is below baseline, then probability is decreased
- The policy parameters are updated based on the reward feedback from the environment

Cont…

- Figure shows results with the gradient-bandit algorithm on a variant of the 10-armed testbed, using normal distribution with a mean of +4 and unit variance
- This shifting up of all the rewards has absolutely no affect on the gradient-bandit algorithm because of the reward baseline term, which instantaneously adapts to the new level
- But if the baseline were omitted, that is, if $\bar{R}_t$ was taken to be constant zero, then performance would be significantly degraded, as shown in the figure



Gradient-bandit algorithms estimate not action values, but action preferences, and favor the more preferred actions in a graded, probabilistic manner using a soft-max distribution

Derivation of gradient-bandit algorithm

In exact *gradient ascent*:

$$H_{t+1}(a) \doteq H_t(a) + \alpha \frac{\partial \mathbb{E}\left[R_t\right]}{\partial H_t(a)}, \tag{1}$$

where:

$$\mathbb{E}[R_t] \doteq \sum_b \pi_t(b) q_*(b),$$

$$\begin{aligned}\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} &= \frac{\partial}{\partial H_t(a)} \left[\sum_b \pi_t(b) q_*(b) \right] \\ &= \sum_b q_*(b) \frac{\partial \pi_t(b)}{\partial H_t(a)} \\ &= \sum_b \left(q_*(b) - X_t \right) \frac{\partial \pi_t(b)}{\partial H_t(a)},\end{aligned}$$

where X_t does not depend on b, because $\sum_b \frac{\partial \pi_t(b)}{\partial H_t(a)} = 0$.

$$
\begin{aligned}
\frac{\partial\, \mathbb{E}[R_t]}{\partial H_t(a)} &= \sum_b \left(q_*(b) - X_t\right) \frac{\partial\, \pi_t(b)}{\partial H_t(a)} \\
&= \sum_b \pi_t(b) \left(q_*(b) - X_t\right) \frac{\partial\, \pi_t(b)}{\partial H_t(a)} / \pi_t(b) \\
&= \mathbb{E}\left[\left(q_*(A_t) - X_t\right) \frac{\partial\, \pi_t(A_t)}{\partial H_t(a)} / \pi_t(A_t) \right] \\
&= \mathbb{E}\left[\left(R_t - \bar{R}_t\right) \frac{\partial\, \pi_t(A_t)}{\partial H_t(a)} / \pi_t(A_t) \right],
\end{aligned}
$$

where here we have chosen $X_t = \bar{R}_t$ and substituted R_t for $q_*(A_t)$, which is permitted because $\mathbb{E}[R_t | A_t] = q_*(A_t)$.
For now assume: $\frac{\partial\, \pi_t(b)}{\partial H_t(a)} = \pi_t(b)\left(\mathbf{1}_{a=b} - \pi_t(a)\right)$. Then:

$$
\begin{aligned}
&= \mathbb{E}\left[\left(R_t - \bar{R}_t\right) \pi_t(A_t) \left(\mathbf{1}_{a=A_t} - \pi_t(a)\right) / \pi_t(A_t)\right] \\
&= \mathbb{E}\left[\left(R_t - \bar{R}_t\right) \left(\mathbf{1}_{a=A_t} - \pi_t(a)\right)\right].
\end{aligned}
$$

$$H_{t+1}(a) = H_t(a) + \alpha\left(R_t - \bar{R}_t\right)\left(\mathbf{1}_{a=A_t} - \pi_t(a)\right), \text{ (from (1), QED)}$$

Thus it remains only to show that

$$\frac{\partial \pi_t(b)}{\partial H_t(a)} = \pi_t(b)(\mathbf{1}_{a=b} - \pi_t(a)).$$

Recall the standard quotient rule for derivatives:

$$\frac{\partial}{\partial x}\left[\frac{f(x)}{g(x)}\right] = \frac{\frac{\partial f(x)}{\partial x} g(x) - f(x)\frac{\partial g(x)}{\partial x}}{g(x)^2}.$$

Using this, we can write...

Quotient Rule: $\frac{\partial}{\partial x}\left[\frac{f(x)}{g(x)}\right] = \frac{\frac{\partial f(x)}{\partial x} g(x) - f(x)\frac{\partial g(x)}{\partial x}}{g(x)^2}$

$$
\begin{aligned}
\frac{\partial \pi_t(b)}{\partial H_t(a)} &= \frac{\partial}{\partial H_t(a)} \pi_t(b) \\
&= \frac{\partial}{\partial H_t(a)} \left[\frac{e^{h_t(b)}}{\sum_{c=1}^{k} e^{h_t(c)}}\right] \\
&= \frac{\frac{\partial e^{h_t(b)}}{\partial H_t(a)} \sum_{c=1}^{k} e^{h_t(c)} - e^{h_t(b)} \frac{\partial \sum_{c=1}^{k} e^{h_t(c)}}{\partial H_t(a)}}{\left(\sum_{c=1}^{k} e^{h_t(c)}\right)^2} && \text{(Q.R.)} \\
&= \frac{\mathbf{1}_{a=b} e^{h_t(a)} \sum_{c=1}^{k} e^{h_t(c)} - e^{h_t(b)} e^{h_t(a)}}{\left(\sum_{c=1}^{k} e^{h_t(c)}\right)^2} && \left(\frac{\partial e^x}{\partial x} = e^x\right) \\
&= \frac{\mathbf{1}_{a=b} e^{h_t(b)}}{\sum_{c=1}^{k} e^{h_t(c)}} - \frac{e^{h_t(b)} e^{h_t(a)}}{\left(\sum_{c=1}^{k} e^{h_t(c)}\right)^2} \\
&= \mathbf{1}_{a=b} \pi_t(b) - \pi_t(b)\pi_t(a) \\
&= \pi_t(b)\left(\mathbf{1}_{a=b} - \pi_t(a)\right). && \text{(Q.E.D.)}
\end{aligned}
$$