Artificial Intelligence
DSE 3252

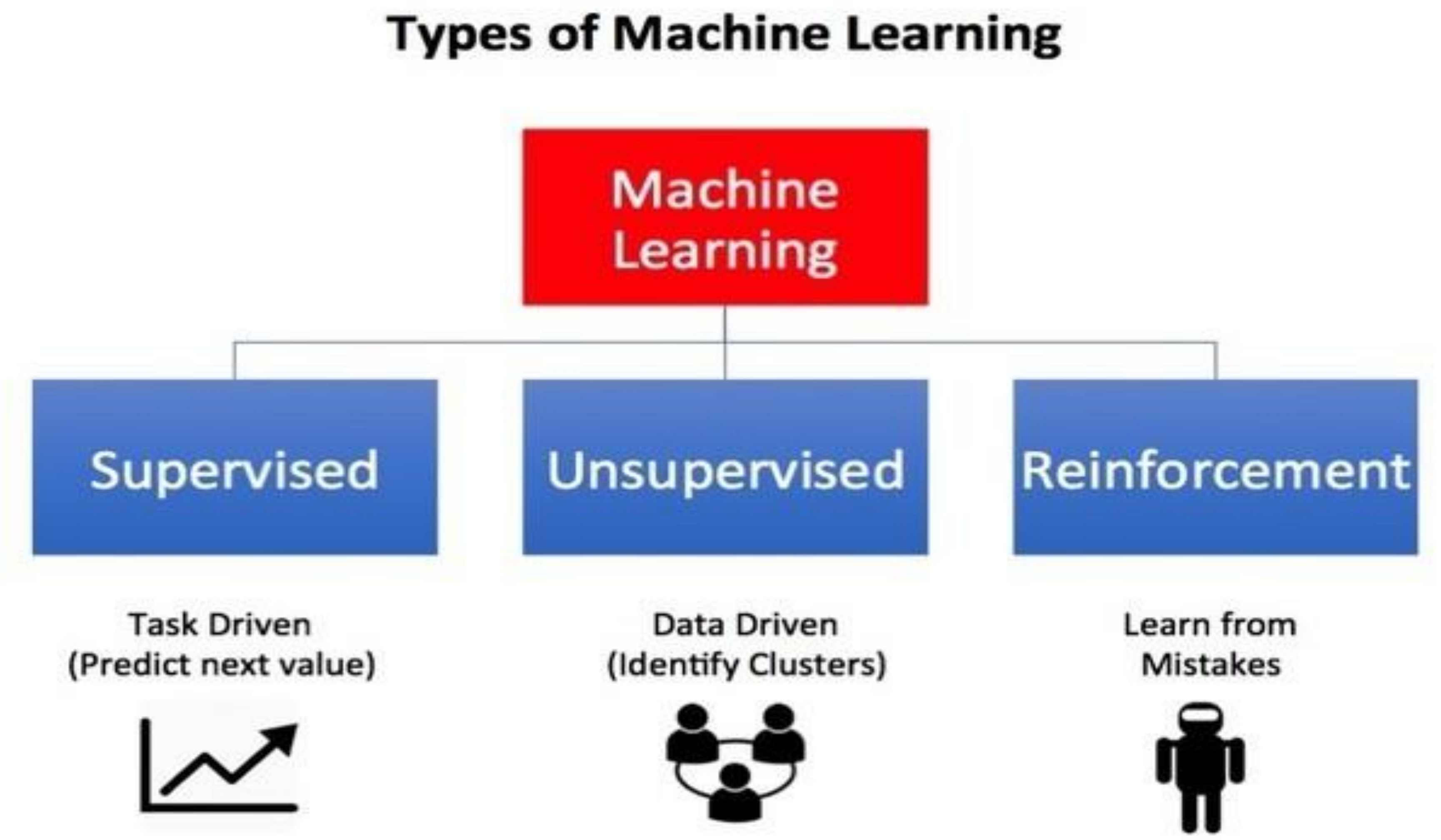
Reinforcement Learning

DR.ROHINI R RAO & DR.RASHMI L MALGHAN
DEPT OF DATA SCIENCE & COMPUTER APPLICATIONS

JANUARY 2024

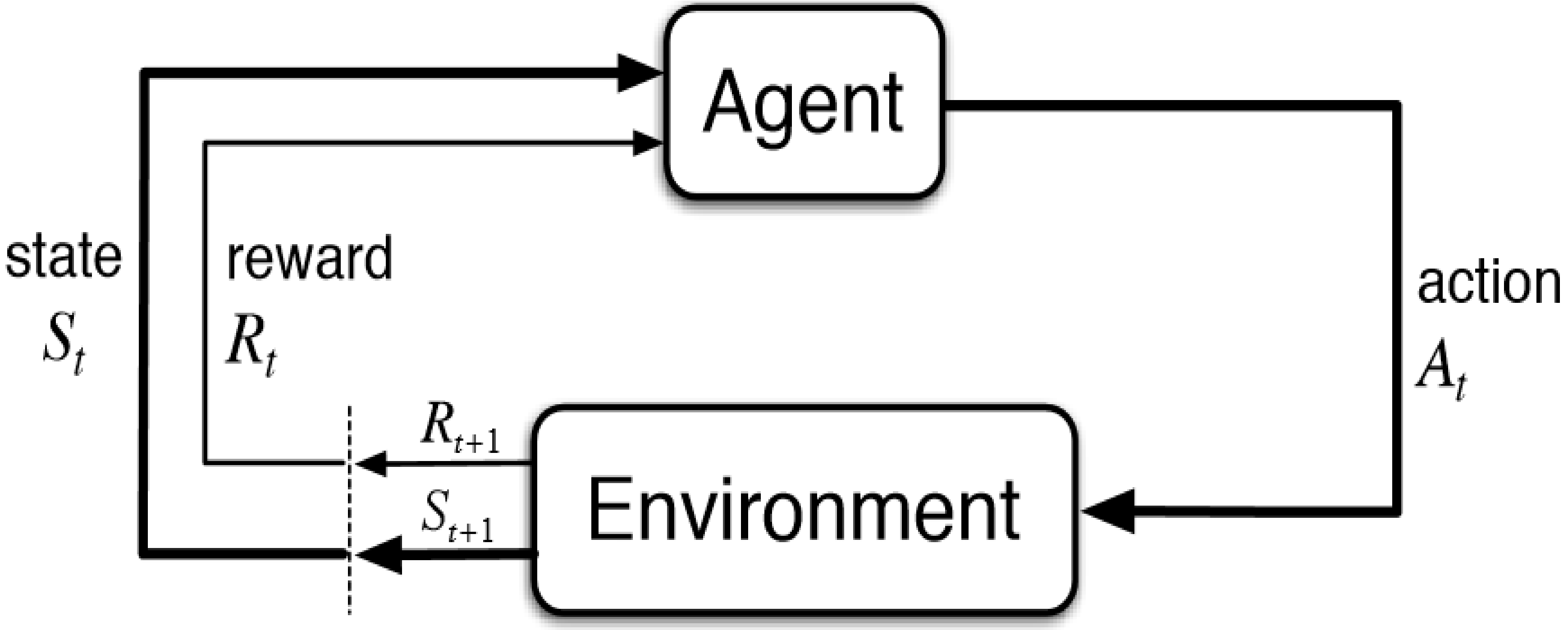
What is Reinforcement Learning

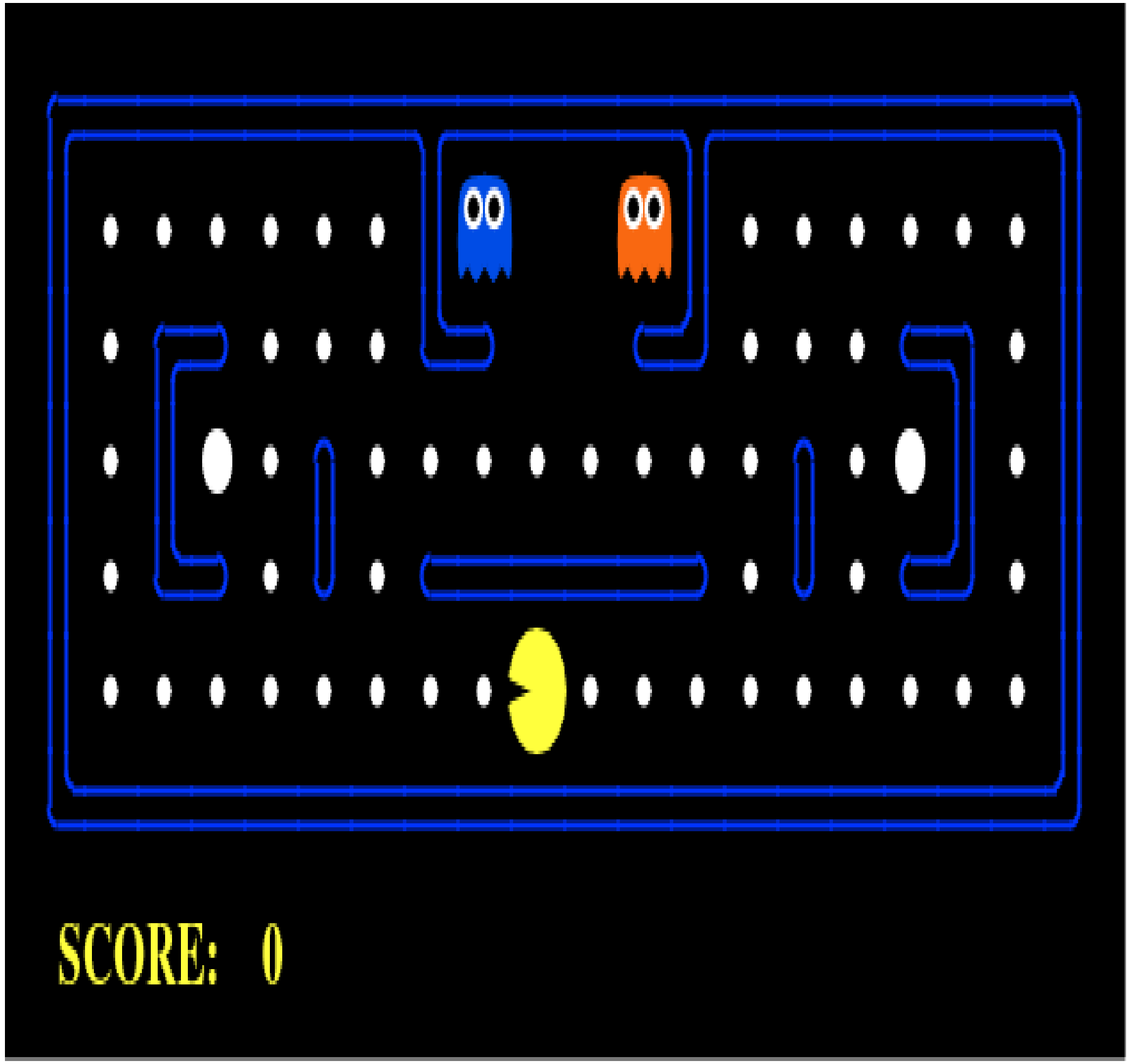
- Reinforcement Learning is a
  - Feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions.
  - For each good action, the agent gets positive feedback
  - for each bad action, the agent gets negative feedback or penalty.
- ***"Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that."***



In Reinforcement learning the goal is to find a suitable action model that would maximize the **total cumulative reward** of the agent

Reinforcement Learning





Problem Formulation in RL

Environment

Physical world in which the agent operates

State

- Current situation of the agent

Reward

- Feedback from the environment

Policy

- Method to map agent's state to actions

Value

- Future reward that an agent would receive by taking an action in a particular state

Markov Decision Process:

The mathematical framework for defining a solution in a reinforcement learning scenario

Can be designed as:

- **Set of states, S**
- **Set of actions, A**
- **Reward function, R**
- **Policy, π**
- **Value, V**
- Set of action (A) has to be taken to transition from our start state to our end state (S).
- Model gets rewards (R) (Positive or Negative) for each action we take
- The set of actions we took defines our policy (π)
- rewards we get in return define our value (V)
- Our task
  - *is to maximize our rewards by choosing the correct policy.*
  - *we have to maximize for all possible values of S for a time t.*

Multi-Armed Bandit Problem

A bandit is defined as someone who steals your money.

One-armed bandit is a simple slot machine wherein you insert a coin into the machine, pull a lever, and get an immediate reward.

A multi-armed bandit

- there are several levers that a gambler can pull, with each lever giving a different return.
- Probability distribution for the reward corresponding to each lever is different and is unknown to the gambler.
- The task is to identify which lever to pull to get maximum reward after a given set of trials.

Multi-Armed Bandit problem (MAB)

- is a special case of **Reinforcement Learning**
- **MAB**
  - collects rewards in an environment by taking some actions after observing some state of the environment.
  - action taken by MAB does not influence the next state of the environment.
  - Therefore, MAB do not model state transitions, credit rewards to past actions, or "plan ahead" to get to reward-rich states.
- Goal of a MAB *agent* is to find a *policy* that collects as much reward as possible.
- **Exploration vs Exploitation Dilemma**
  - Not a good idea to exploit the action that promises the highest reward
  - because then there is a chance that we miss out on better actions if we do not explore enough.

Value of Action

In our k-armed bandit problem, each of the k actions has an expected or mean reward given that that action is selected

- A_t - action selected on time step t
- R_t- corresponding reward as Rt.
- The value of an arbitrary action a, denoted $q_*(a)$
- the expected reward given that a is selected: $q_*(a) = E[R_t \mid A_t = a]$

If you knew the value of each action, then it would be trivial to solve the k-armed bandit

Problem

- $Q_t(a)$ - the estimated value of action a at time step t
- $Q_t(a)$ to be close to $q_*(a)$

Exploitation

- maximize the expected reward on the one step
- maintain estimates of the action values
- Greedy action - at any time step there is at least one action whose estimated value is greatest

Exploration

- exploration may produce the greater total reward in the long run
- greedy action’s value is known with certainty, while several other actions are estimated to be nearly as good but with substantial uncertainty

Action Value Methods

Sample Average Method

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

where $1_{predicate}$ denotes the random variable that is 1 if predicate is true and 0 if it is not

If the denominator

- is 0 - define $Q_t(a)$ as some default value, such as 0.
- goes to infinity- by the law of large number, Qt(a) converges to $q_*(a)$

Greedy action selection method

$$A_t \doteq \underset{a}{\arg\max}\, Q_t(a)$$

- argmax_a denotes the action a for which the expression that follows is maximized
- Variation - -greedy - select randomly from among all the actions with equal probability

exploration vs exploitation dilemma

Arm	Reward
1	0
2	0
3	1
4	1
5	0
3	1
3	1
2	0
1	1
4	0
2	0

Pure exploitation approach

- select only one slot machine and keep pulling the lever all day long.
- may give you “some” payouts.
- might hit the jackpot (with a probability close to 0.00000….1)

Pure exploration approach

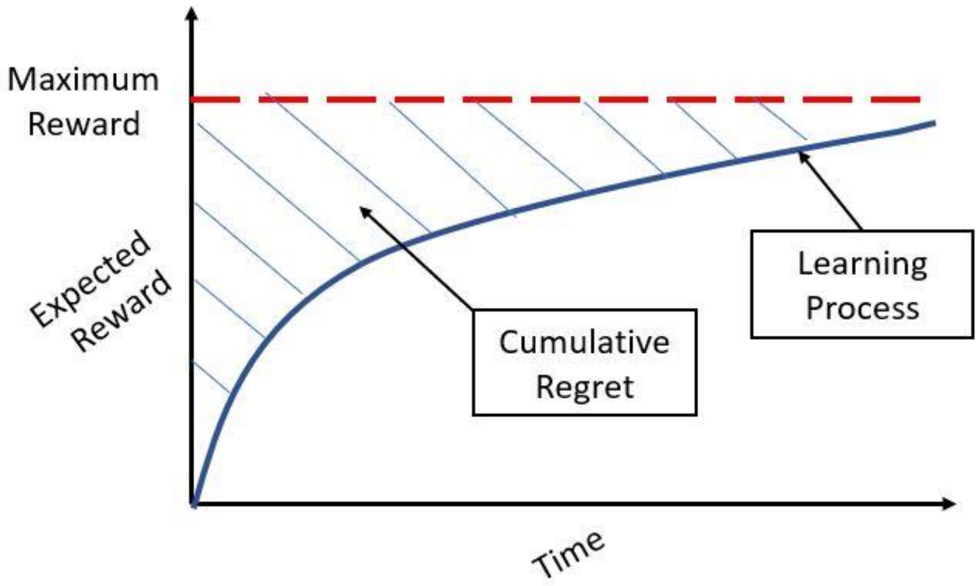
- pull a lever of each & every slot machine
- Get sub-optimal payouts
- May be at least one of them would hit the jackpot.

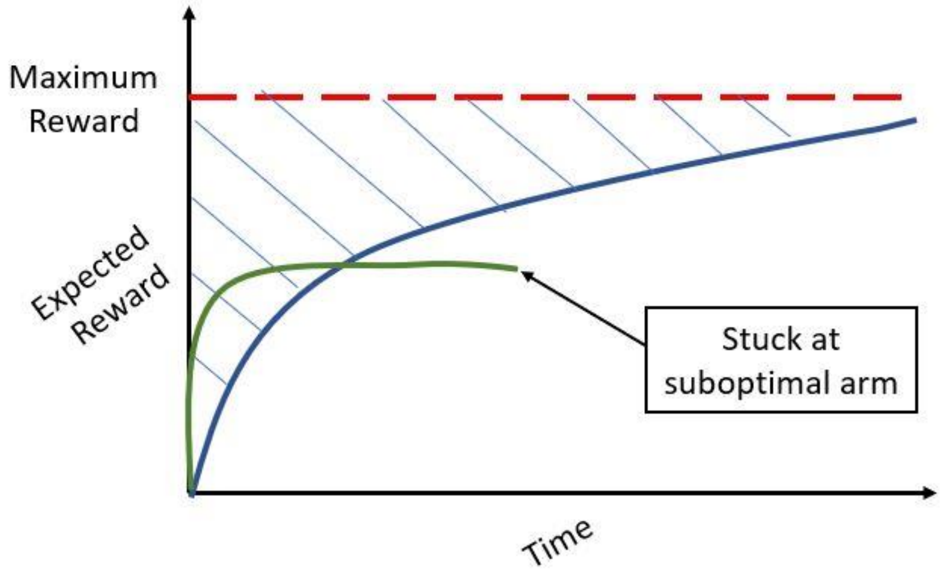
Exploration vs Exploitation trade-off

- To build an optimal policy, the agent faces the dilemma of exploring new states while maximizing its overall reward at the same time.

The best overall strategy may involve short-term sacrifices.

Therefore, the agent should collect enough information to make the best overall decision in the future.





Exploitation vs Exploration

ε -Greedy Method

- Behave greedily most of the time
- A few times with small probability Epsilon, select randomly from among all the actions with equal probability
- independently of the action-value estimates.

Advantages

- 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 $q_*(a)$

Incremental Implementation

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

- Error in the estimate – [Target−OldEstimate]
- Target is the n^{th} reward.
- StepSize changes from time step to time step
- In processing the n^{th} reward for action a, the method uses the step-size parameter 1/n
- step size parameter is denoted as $\alpha_t(a)$
- Update Rule

$$
NewEstimate \leftarrow OldEstimate \;+\; StepSize\Big[Target - OldEstimate\Big]
$$

Simple Bandits Algorithm

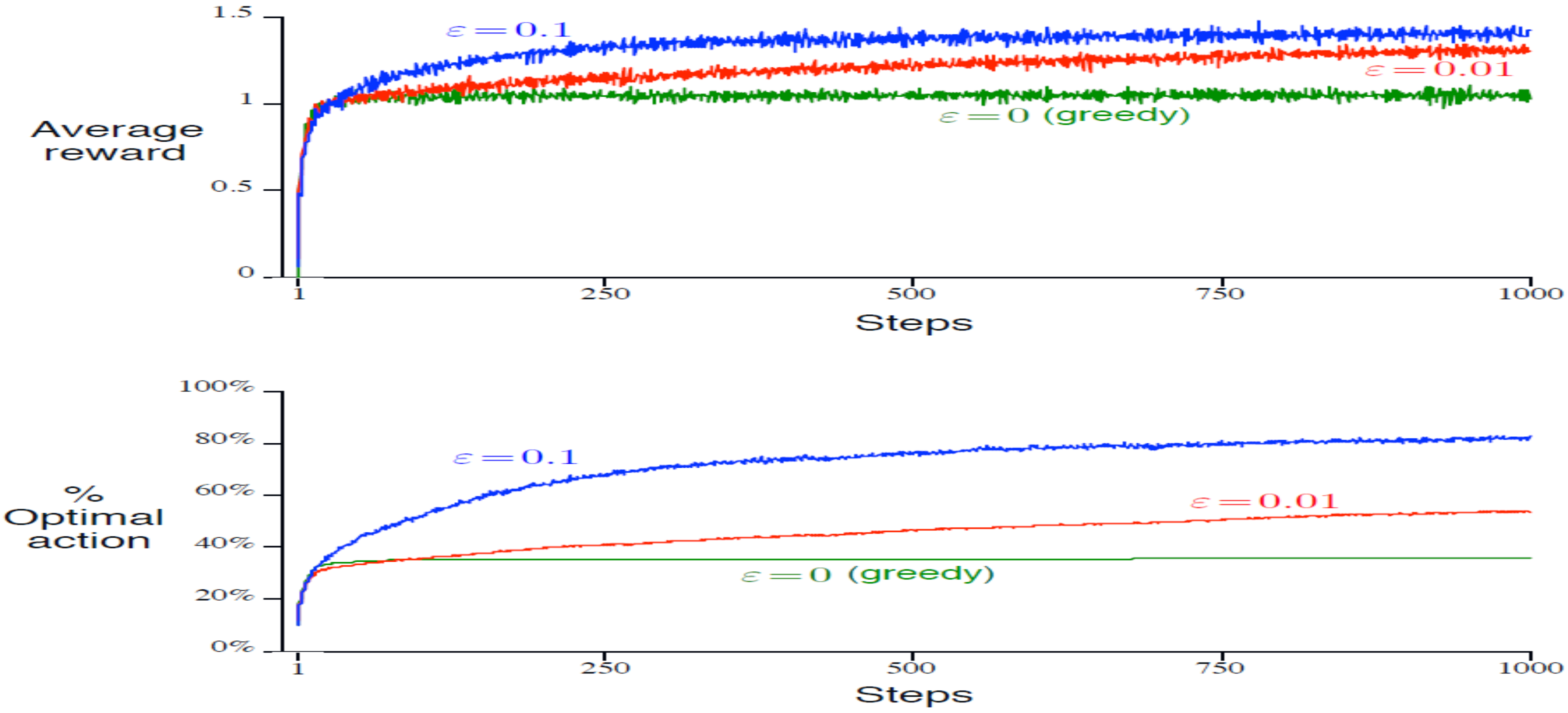
Initialize, for $a = 1$ to k:

$$Q(a) \leftarrow 0$$
$$N(a) \leftarrow 0$$

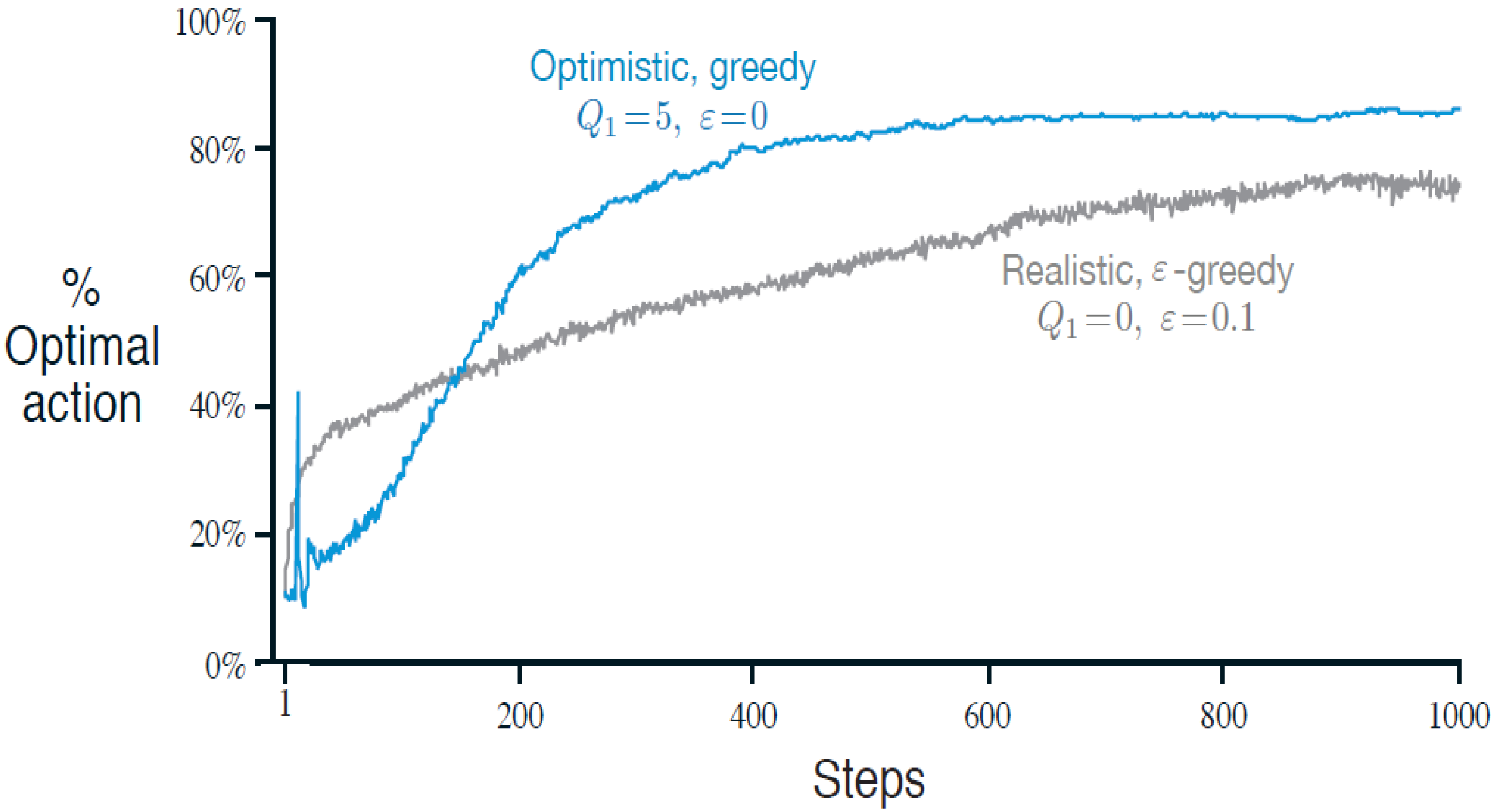
Loop forever:

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

Average Performance of ε -greedy



The effect of optimistic initial action values



Upper-confidence-bound action selection

Optimism in the face of uncertainty

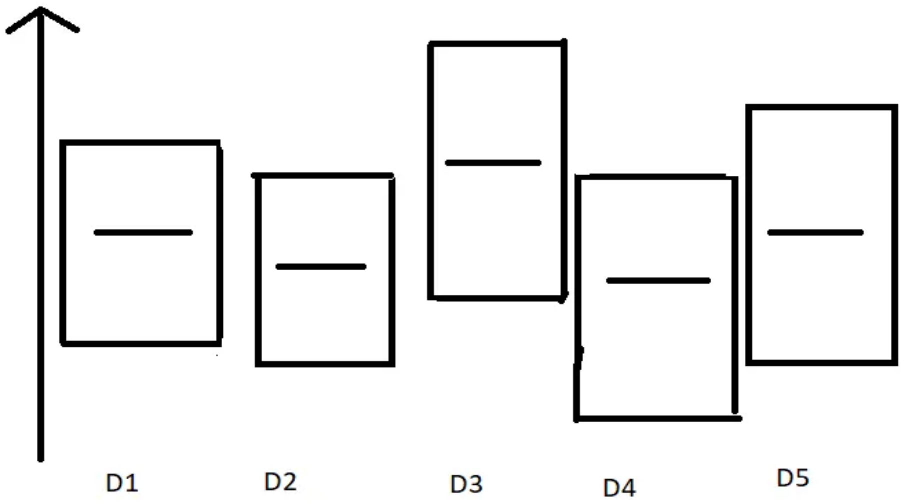
$$A_t \doteq \underset{a}{\operatorname{argmax}} \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right],$$

where;

•*$Q_t(a)$ is the estimated value of action 'a' at time step 't'.*

•*$N_t(a)$ is the number of times that action 'a' has been selected, prior to time 't'.*

•*'c' is a confidence value that controls the level of exploration.*



UCB Action Selection

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

Exploitation:

- $Q_t(a)$ represents the exploitation
- if you don't know which action is best then choose the one that currently looks to be the best

- ***Exploration:***
  - If an action hasn't been tried very often, or not at all, then $N_t(a)$ will be small. Consequently, the uncertainty term will be large, making this action to be selected
  - *As* $N_t(a)$ increments, and the uncertainty term decreases, making it less likely that this action will be selected as a result of exploration
  - it may still be selected as the action with the highest value

As n goes to infinity the exploration term gradually decreases until eventually, actions are selected based only on the exploitation term.

Steps followed in UCB agent

1. *At each round **t**, we compute two numbers for arm A.*
   -> $N_A(t)$ = number of times the arm A was selected up to round t.
   -> $R_A(t)$ = number of rewards of the arm A up to round t.
2. *From these two numbers we have to calculate,*
   a. The average reward of machine m up to round t

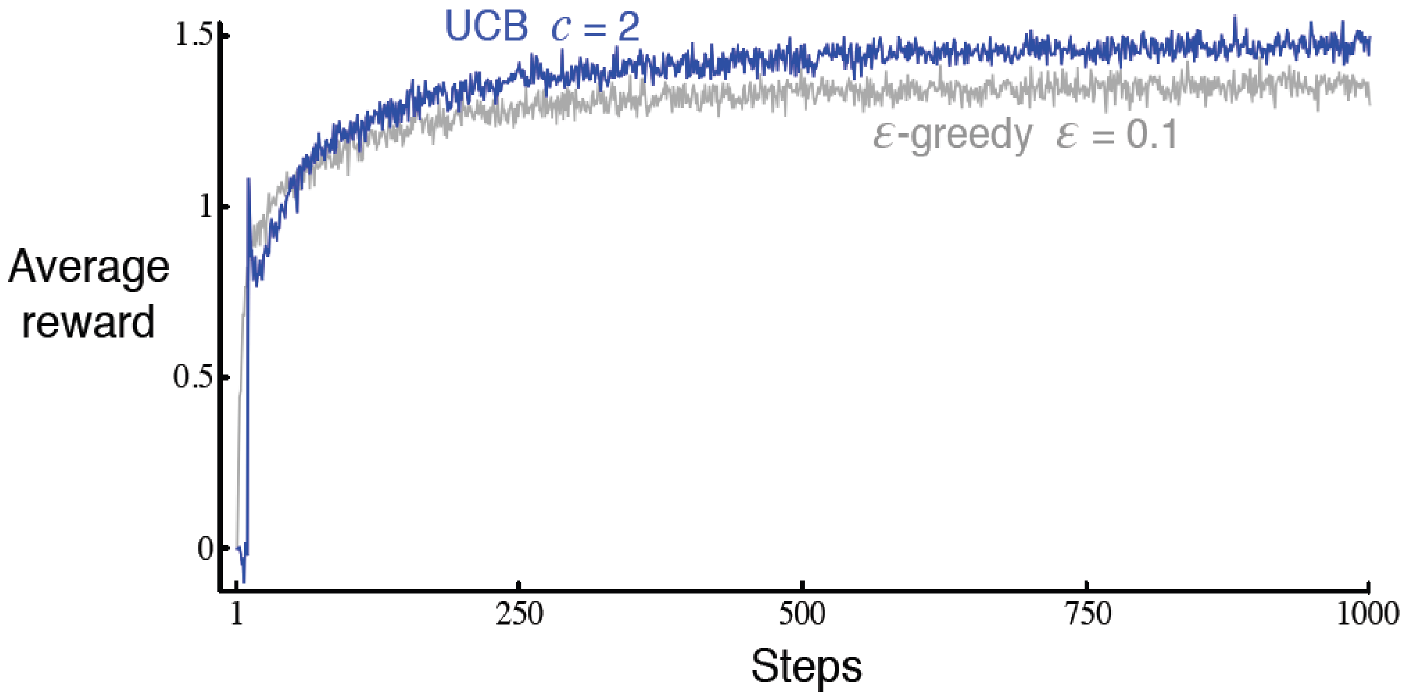
$$r_A(t) = R_A(t) \,/\, N_A(t).$$

b. The confidence interval [$r_A(t)$ — $\Delta_A(t)$, $r_A(t)+\Delta_A(t)$] at round n

with, $\Delta_A(t) \Rightarrow c * \mathrm{sqrt}(\ln(t) \,/\, N_a(t))$

1. *We select the arm A that has the maximum UCB,* ($r_A(t)+\Delta_A(t)$)

Upper-confidence-bound action selection



Gradient Bandit Algorithms

Probability of taking action a at time t

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

The action preferences are updated by

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

Initially all action preferences are the same (e.g., $H_1(a) = 0$, for all a)

As Stochastic Gradient Ascent

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

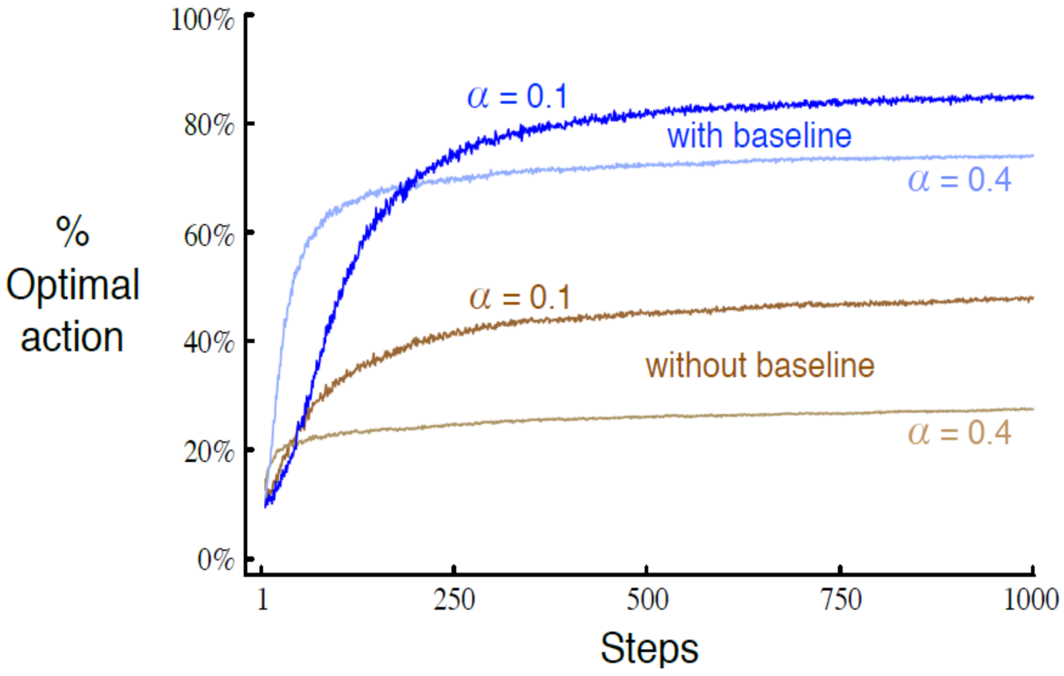
$$\mathbb{E}[R_t] = \sum_x \pi_t(x) q_*(x)$$

Where
α> 0 is step size parameter

$$\bar{R}_t \in \mathbb{R}$$

Is average of all rewards upto and including time t

Gradient Bandit Algorithms



Without Baseline –

$\bar{R}_t$ is set to 0

Ad Optimization

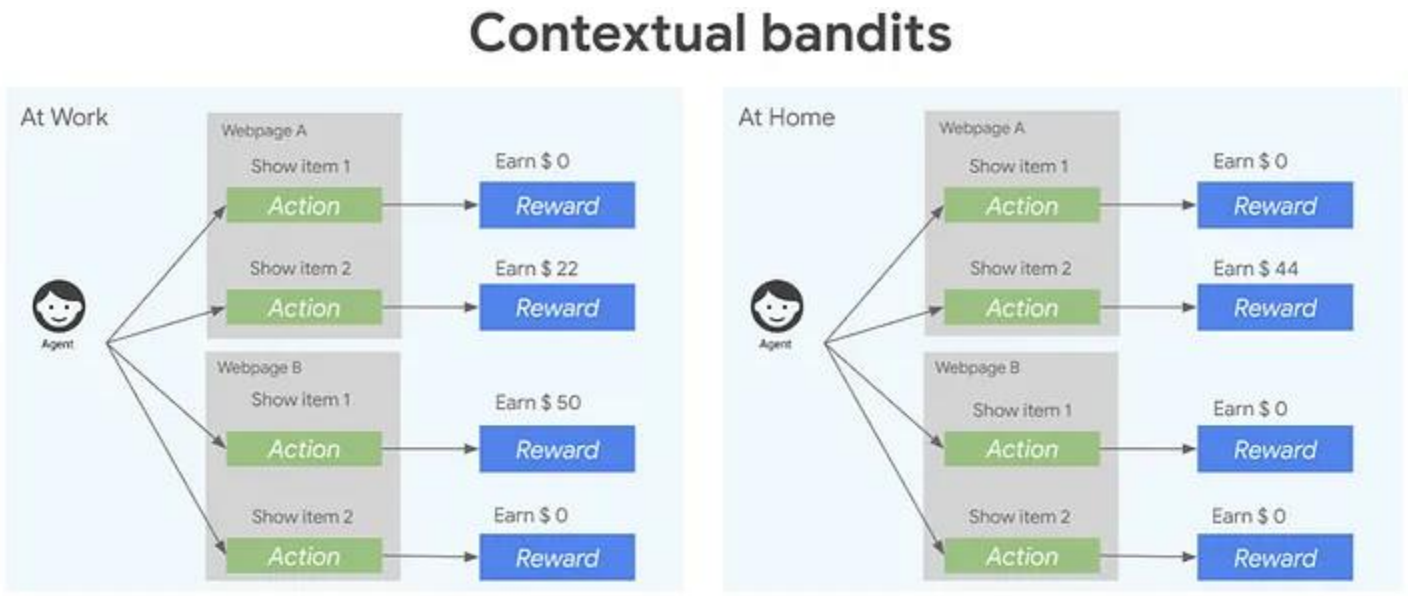
Ad 1	Ad 2	Ad 3	Ad 4	Ad 5	Ad 6	Ad 7	Ad 8	Ad 9	Ad 10
1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0

Associative Search (Contextual Bandits)

In k-armed bandit problem each action affects only the immediate reward

Contextual Bandits

- actions are affected by **context, which in turn affects the reward**
- In a general RL task, there is more than one situation/context
- **the goal is to learn a policy**: a mapping from situations/context to the actions that are best in those situations



Multi-armed Bandits Problem

- K actions (feature-free)
- Each action has an average reward (unknown): μ_k
- For t=1,…,T (unknown)
  - Choose an action a_t from {1, …, K} actions
  - Observe a random reward $y_{t,}$ where y_t is bounded [0,1]
  - $E[y_t] = \mu_{a,t}$: Expected reward of action a_t
- Minimizing Regret: $$R = \sum_{t=1}^{T} [\mu^* - \mu_{a,t}]$$

regret is the difference between the total reward achieved by always selecting the optimal arm and the total reward achieved by the algorithm.

Q. How to choose an action to minimize regret?

Feature free bandit setting

Users u_1 with age YOUNG and u_2 with age OLD

u_1

u_2

Retirement planning wishes vs. reality

The Player Wizarding World of Harry Potter ride may conjure a new path for theme park rides

Elon Musk: 198,000 Tesla Model 3 Orders Received in 24 Hours

Not tired yet: Warriors top Spurs for 72nd win, set up date with history

Contextual Bandit Problem

- For t=1,…,T (unknown)
  - User u_t , set A_t of actions (a)
  - Feature vector (context) $\mathbf{x}_{t,a}$: summarizes both user u_t and action a
  - Based on previous results, choose a_t from A_t
  - Receive payoff r_{t,a_t}
  - Improve selection strategy with new observation set $(x_{t,a_t}, a_t, r_{t,a_t})$

$$\mathbf{E}[r_{t,a}|\mathbf{x}_{t,a}] = \mathbf{x}_{t,a}^{\top}\boldsymbol{\theta}_a^*.$$

Minimizing Regret: $R(T) = \mathbf{E}\left[\sum_{t=1}^{T}\left(r_{t,a_t^*} - r_{t,a_t}\right)\right]$

Action with maximum expected payoff at time t

Contextual Linear Bandits setting

Linear Payoff = $x^T \theta$

Users u_1 with age YOUNG and u_2 with age OLD

$$x_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$x_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

[0.1 , 0.6] **Retirement planning wishes vs. reality**

[0.5, 0.1] The Player **Wizarding World of Harry Potter ride may conjure a new path for theme park rides**

[0.6 , 0.1] **Elon Musk: 198,000 Tesla Model 3 Orders Received in 24 Hours**

[0.9 , 0.2] **Not tired yet: Warriors top Spurs for 72nd win, set up date with history**

Contextual Bandit

For each trail t=1,2,3..., T

1. Observe environment $x_{t,a} \in \mathbb{R}^d$, i.e. user u_t a set of actions $\mathcal{A}_t$ and both their features
2. Choose an arm $a_t \in \mathcal{A}$ based on previous trails an receive payoff r_{t,a_t}.
3. Improve arm selection strategy with new observation $(\mathrm{x}_{t,a_t}, a_t, r_{t,a_t})$

Example: News Recommendation

For each time the news page is loaded t=1,2,3..., T

1. Arms or actions are the articles, which can be shown to the user. The environment could be user and article information.
2. If the aricle is clicked $r_{t,a_t} = 1$ otherwise 0.
3. Improve new article selection

Minimize expected regret, i.e

$$R_A(T) = \mathbb{E}\left[\sum_{t=1}^{T} r_{t,a_t^*}\right] - \mathbb{E}\left[\sum_{t=1}^{T} r_{t,a_t}\right]$$

Linear Disjoint Model

- *disjoint* since the parameters are not shared among different arms.
- To solve for the coefficient vector Θ ridge regression is applied to the training data.

$$E[r_{t,a} | x_{t,a}] = [x_{t,a}]^T \theta_a^*$$

- How to estimate θ_a?
  - Linear regression solution to θ_a is

$$\widehat{\boldsymbol{\theta_a}} = \boldsymbol{argmin_\theta} \sum_{m \in \boldsymbol{D_a}} ([x_{t,a}]^T \theta_a - \boldsymbol{b_a^{(m)}})^2$$

We can get:

$$\widehat{\boldsymbol{\theta_a}} = (\boldsymbol{D_a^T D_a} + \boldsymbol{I_d})^{-1} \boldsymbol{D_a^T b_a}$$

$\boldsymbol{D_a}$ is a m × d matrix of **m** training inputs $[x_{t,a}]$

$\boldsymbol{b_a}$ is a m-dimension vector of responses to $\boldsymbol{a}$(click/no-click)

linUCB Algorithm

- Initialization: $A_a \stackrel{\text{def}}{=} \boldsymbol{D_a^T D_a} + \boldsymbol{I_d}$
  - For each arm a:
    - $A_a = I_d$ //identity matrix d ×d
    - $b_a = [0]_d$ //vector of zeros
- Online algorithm:
  - For t=[1:T]:
    - Observe features for all arms a : $x_{t,a} \in R^d$
    - For each arm a :
      - $\theta_a = A_a^{-1} b_a$ //regression coefficients
      - $p_{t,a} = [x_{t,a}]^T \theta_a + \boldsymbol{\alpha}\sqrt{[x_{t,a}]^T A_a^{-1} x_{t,a}}$
    - Choose arm $a_t = argmax_a p_{t,a}$ //choose arm
    - $A_{a_t} = A_{a_t} + x_{t,a_t}[x_{t,a_t}]^T$ //update A for the chosen arm a_t
    - $b_{a_t} = b_{a_t} + r_t\, x_{t,a_t}$ //update b for the chosen arm a_t

Thomson Sampling

Basic Intuition

1. all machines are assumed to have a uniform distribution of the probability of reward
2. For each observation, a new distribution of rewards is generated (exploration)
3. Further observations are used to update the success distributions of rewards
4. After sufficient observations, each slot machine will have a success distribution of rewards (exploitation)

- A simple natural Bayesian heuristic
  - Maintain a belief(distribution) for the unknown parameters
  - Each time, pull arm a and observe a reward r

- Initialize priors using belief distribution
  - For t=1:T:
    - Sample random variable X from each arm's belief distribution
    - Select the arm with largest X
    - Observe the result of selected arm
    - Update prior belief distribution for selected arm

Example: Web Content Personalization

Vowpal Wabbit

- an interactive ML library and the RL framework for services like Microsoft Personalizer.
- It allows for maximum throughput and lowest latency when making personalization ranks and training the model with all events

Con-Ban Agent performs the following functions:

Some context ‘x’ arrives and is observed by Con-Ban Agent.

Con-Ban Agent chooses an action ‘a’ from a set of actions A, i.e., $a \in A$ (A may depend on ‘x’).

Some reward ‘r’ for the chosen ‘a’ is observed by Con-Ban Agent.

For example: **Con-Ban Agent news website**:

- **Decision to optimize:** articles to display to user.
- **Context:** user data (browsing history, location, device, time of day)
- **Actions:** available news articles
- **Reward:** user engagement (click or no click)

Summary of Notation

$\doteq$	equality relationship that is true by definition
$\approx$	approximately equal
$\propto$	proportional to
$\Pr\{X = x\}$	probability that a random variable X takes on the value x
$X \sim p$	random variable X selected from distribution $p(x) \doteq \Pr\{X = x\}$
$\mathbb{E}[X]$	expectation of a random variable X, i.e., $\mathbb{E}[X] \doteq \sum_x p(x)x$
$\arg\max_a f(a)$	a value of a at which $f(a)$ takes its maximal value
$\ln x$	natural logarithm of x
e^x	the base of the natural logarithm, $e \approx 2.71828$, carried to power x; $e^{\ln x} = x$
$\mathbb{R}$	set of real numbers
$f : \mathcal{X} \rightarrow \mathcal{Y}$	function f from elements of set $\mathcal{X}$ to elements of set $\mathcal{Y}$
$\leftarrow$	assignment
$(a, b]$	the real interval between a and b including b but not including a
ε	probability of taking a random action in an ε-greedy policy
α, β	step-size parameters
γ	discount-rate parameter
λ	decay-rate parameter for eligibility traces
$\mathbb{1}_{predicate}$	indicator function ($\mathbb{1}_{predicate} \doteq 1$ if the *predicate* is true, else 0)

In a multi-arm bandit problem:

k	number of actions (arms)
t	discrete time step or play number
$q_*(a)$	true value (expected reward) of action a
$Q_t(a)$	estimate at time t of $q_*(a)$
$N_t(a)$	number of times action a has been selected up prior to time t
$H_t(a)$	learned preference for selecting action a at time t
$\pi_t(a)$	probability of selecting action a at time t
$\bar{R}_t$	estimate at time t of the expected reward given π_t

References

1. Richard S Sutton, Andrew G Barto, Reinforcement Learning, second edition, MIT Press
2. Russell S., and Norvig P., Artificial Intelligence A Modern Approach (3e), Pearson 2010
3. https://www.coursera.org/learn/fundamentals-of-reinforcement-learning/home/week/1