REINFORCEMENT LEARNING

FUNDAMENTALS
+
APPLICATIONS

WEEK 1

BANDIT ALGORITHM

WEEK 1: CORE CONCEPTS

State: A representation of the **agent**'s current situation. This can be a simple ID (State 1, State 2, …), coordinates, or a set of features. It can include compressed representations of the past (i.e., memory).

Action: Selected by the **Agent**, according to their Policy, that affect the Environment.

Reward: The special learning signal emitted from the Environment in response to **Agent** actions. Used by the **Agent** to learn about which actions and states are "good". Rewards are usually specified by you.

Timestep: For this week, timesteps represent discrete action-reward-update cycles. Our learner takes an action, receives some reward, and updates its value estimates. Then we go to the next timestep.

Run: One run of N timesteps. We might conduct 1000 runs of 2000 timesteps.

WEEK 1: CORE CONCEPTS

Learning Rate: Represented by the symbol $\alpha \in [0,1]$, learning rate is the weight on the most recent reward.

Policy: Represented by the symbol π, policies are how agents choose their actions. An example policy might be "always choose the action that I think gives me the best reward RIGHT NOW." Examples of policies from this week are epsilon-greedy and Upper-Confidence-Bound.

Bandit

Any Non-Lawful

Challenge: 1/8

(25 XP)

Medium Humanoid (Any Race)

Speed: 30'

AC: 12 HP: 11

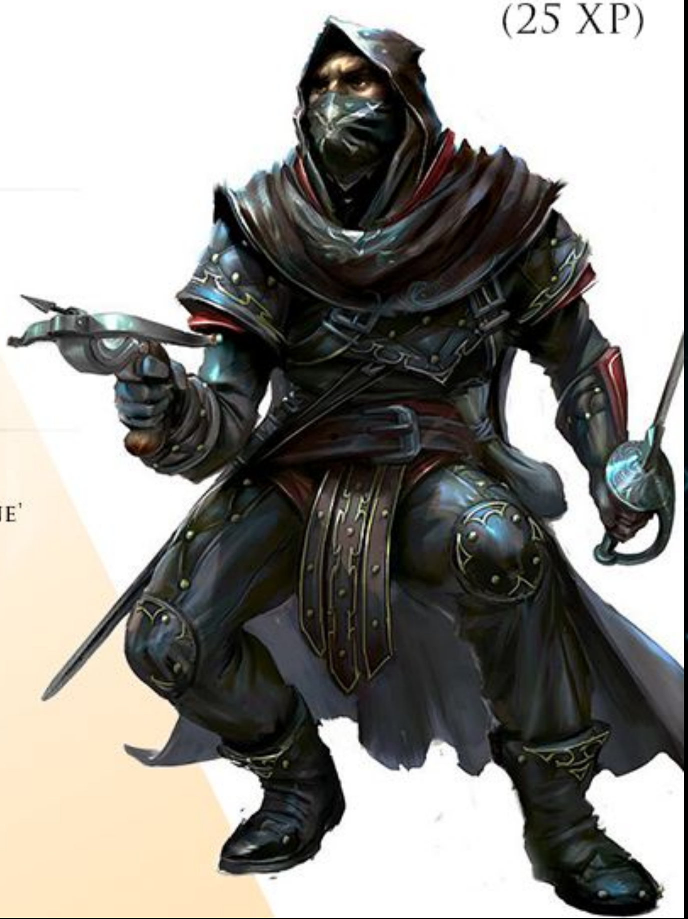
STR	DEX	CON	INT	WIS	CHA
11 (+0)	12 (+1)	12 (+1)	10 (+0)	10 (+0)	10 (+0)

Senses:	Passive Perception 10
Languages:	Any one language (Usually Common)

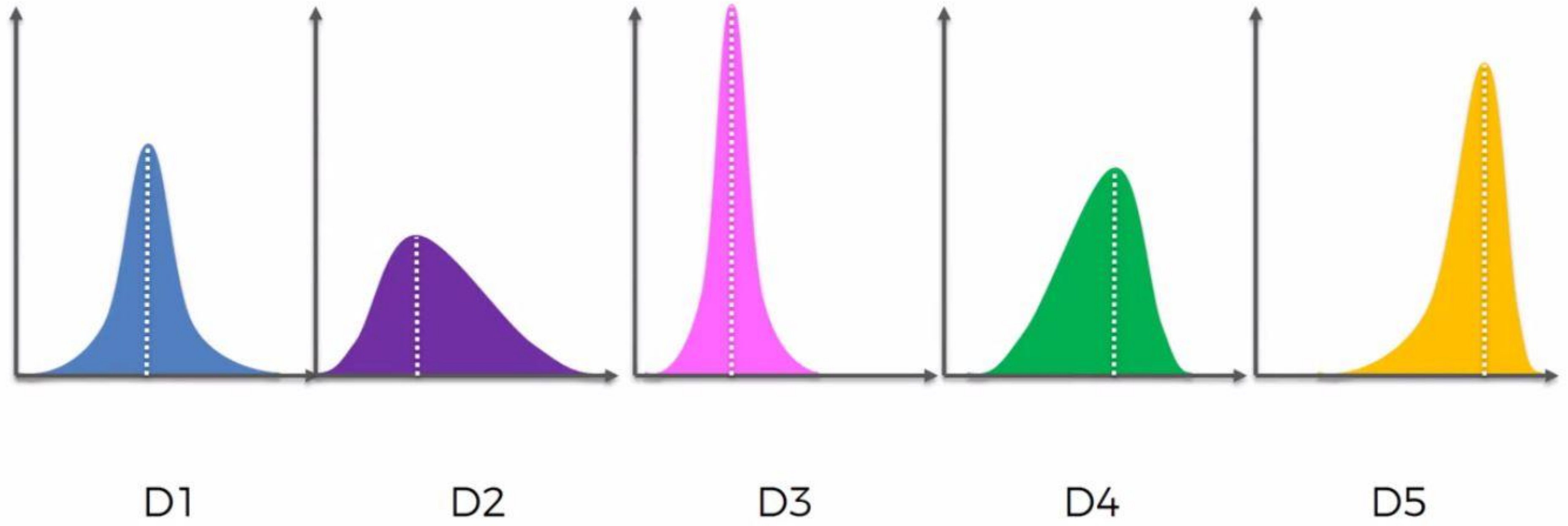
Actions:

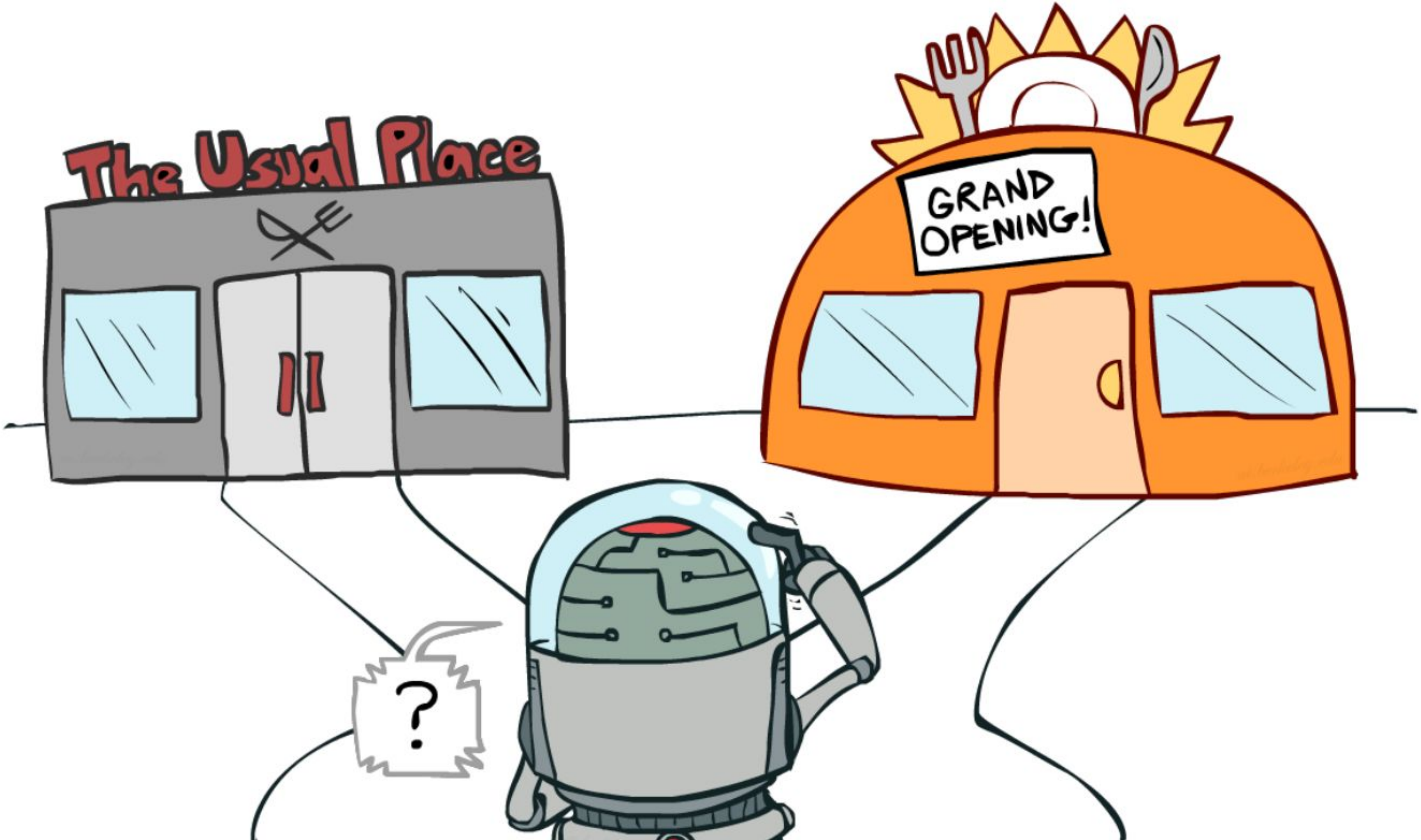
Scimitar.
Melee Weapon Attack: +3 to hit, reach 5 ft., one' target. Hit: 4 (1d6 + 1) slashing damage.

Light Crossbow.
Ranged Weapon Attack: +3 to hit, range 80 ft./320ft., one target. Hit: 5 (1d8 + 1) piercing damage.

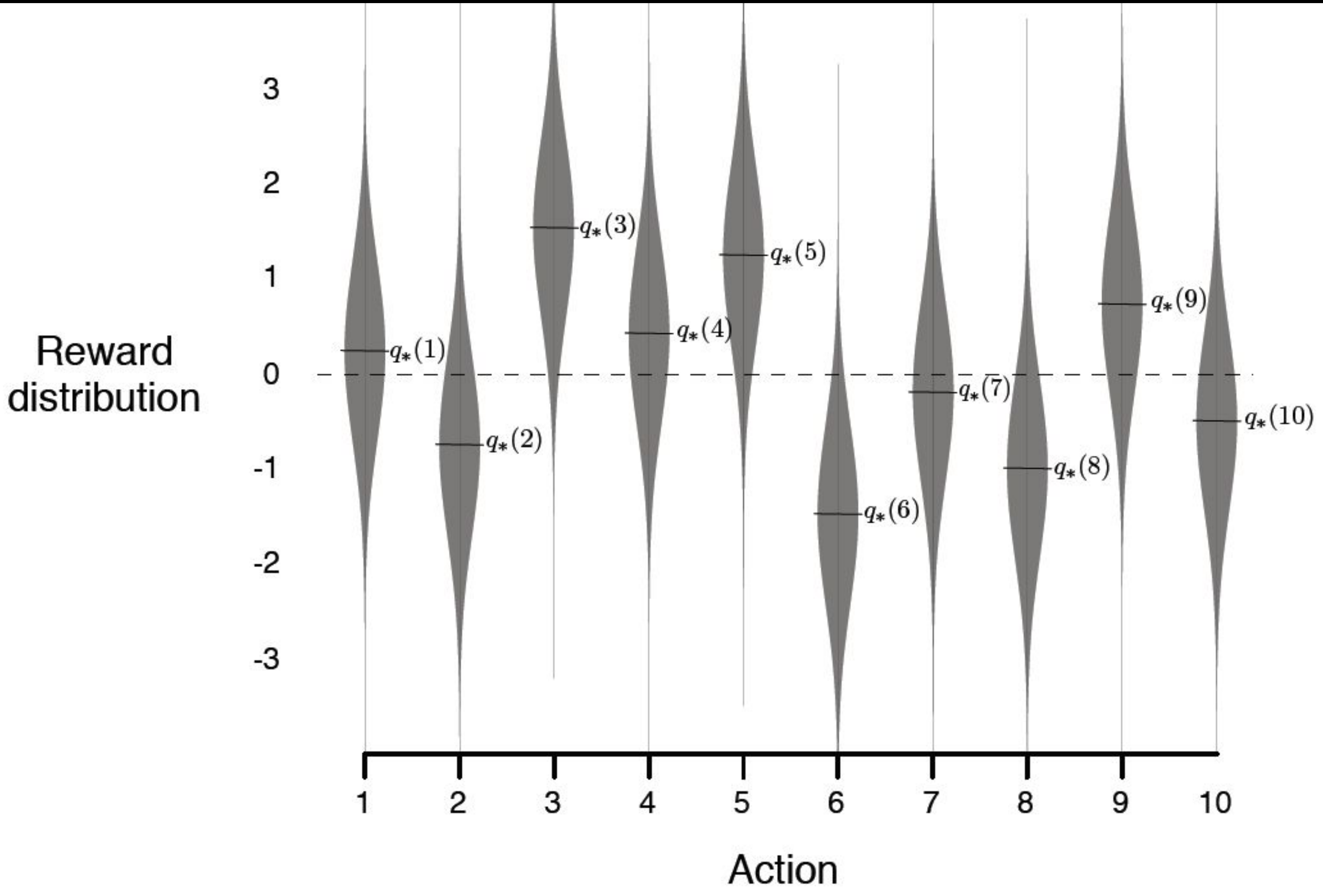


The Multi-Armed Bandit Problem




The Usual Place
GRAND OPENING!
?

LEFT
EXIT 12
GETTING ADEQUATE REST
SEEING HOW MANY BEERS I CAN CRAM IN AFTER DAY SHIFT
CONSISTENTLY ME



Reward distribution
3
2
1
0
-1
-2
-3
$q_*(1)$
$q_*(2)$
$q_*(3)$
$q_*(4)$
$q_*(5)$
$q_*(6)$
$q_*(7)$
$q_*(8)$
$q_*(9)$
$q_*(10)$
1
2
3
4
5
6
7
8
9
10
Action

EQUATION : ACTION-VALUE

The estimated value of action a at timestep *t*.

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

“The **value** of arbitrary **action *a***, denoted by $q_*(a)$, is the **expected reward *r*** given that ***a*** was selected.”

EQUATION : ACTION-VALUE

The estimated value of action a at timestep *t*.

$$Q_t(a) = \frac{\text{sum of rewards when } a \text{ taken prior to current timestep } t}{\text{number of times } a \text{ taken prior to timestep } t}$$

EQUATION : ACTION-VALUE

The estimated value of action a at timestep *t*.

$$Q_t(a) = \frac{1}{N(a)} \sum_{i=1}^{n-1} R_i$$

$Q_t(a)$: The estimated value of action a at timestep *t*.

$N(a)$: The number of times action a was taken prior to timestep *t*.

R_i : The reward received at prior timestep *i*.

EQUATION : ACTION-VALUE

INCREMENTAL UPDATE, SAMPLE AVERAGE

$$Q_{n+1} = Q_n + \frac{1}{n} [R_n - Q_n]$$

ALGORITHM : SIMPLE BANDIT

```
Initialization: $Q(a) \longleftarrow 0$, $N(a) \longleftarrow 0$, $k \in \mathbb{Z}^{0+}$, $N \in \mathbb{Z}^{0+}$

For $t = 1, 2, ..., n$ DO:
    $A \longleftarrow \begin{cases} \text{argmax}_a Q(a) \text{ with probability } 1 - \varepsilon \\ \text{Random action with probability } \varepsilon \end{cases}$
    Take action $a$, receive reward $R$
    $N(a) = N(a) + 1$
    $Q(a) = Q(a) + \frac{1}{N(a)} [\, R - Q(a) \,]$
```

EQUATION : ACTION-VALUE

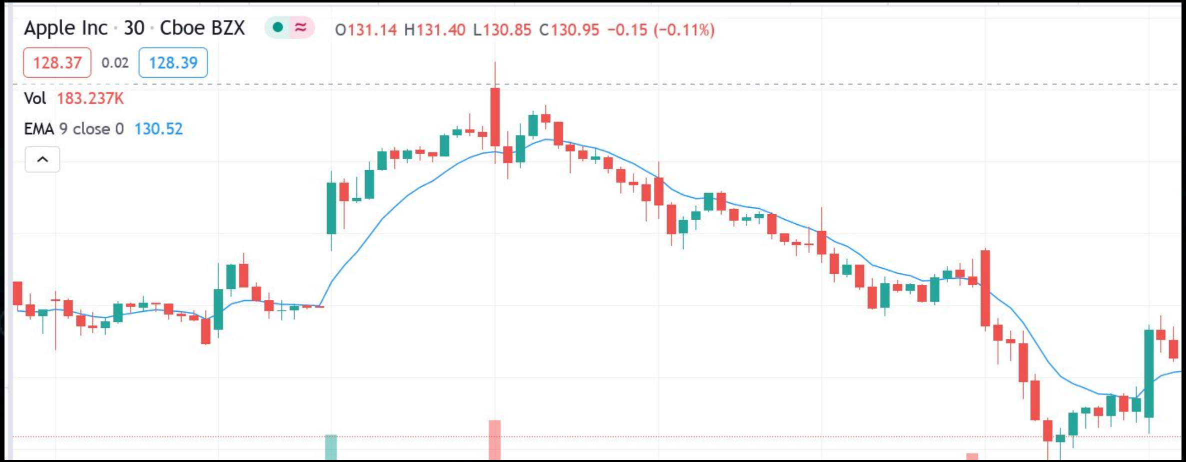
EXPONENTIAL RECENCY-WEIGHTED AVERAGE

$$Q_{n+1} = \alpha[R_n - Q_n]$$

α : Learning rate.

EQUATION : ACTION-VALUE

EXPONENTIAL RECENCY-WEIGHTED AVERAGE

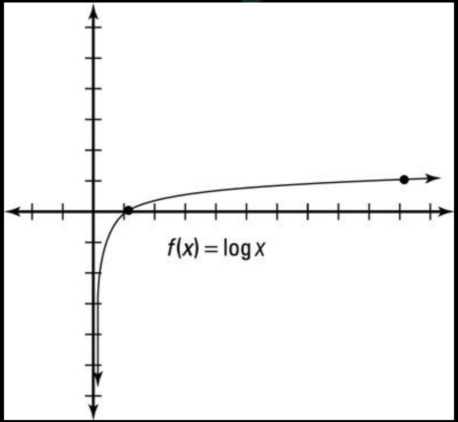


A exponential recency-weighted average (blue line) applied to price action of Apple stock. Note that this is a *lagging indicator*: it only responds to shifts after some delay (lag). The lower α is, the more it will lag. Sometimes we want a larger lag so our learner is not overly reactive to extreme draws or short-lived shifts in distribution. Sometimes we want a shorter lag so our learner can react quickly to sudden shifts in distribution.

EQUATION : UCB

UPPER CONFIDENCE BOUND π

$$A_t = \text{argmax}\, Q_t(a) + c[\sqrt{\ } log(t) / N_t(a)\ ]$$



GRADIENT BANDIT

PREFERENCE UPDATE

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \overline{R}_t)(1 - \pi_t(A_t))$$

$$H_{t+1}(a) = H_t(a) - \alpha(R_t - \overline{R}_t)\pi_t(a), \ \forall\ a \neq A_t$$

$H_t(a)$ preference value for action a
α : Learning rate
R_t : Reward received at time *t*.
R_t bar : Average reward received up to and including time *t*.

$\pi_t(a_t)$: The probability [0,1] of choosing action a at time t, as obtained by passing our preference values H through a softmax function.

GRADIENT BANDIT

PREFERENCE UPDATE

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \overline{R}_t)(1 - \pi_t(A_t))$$

$$H_{t+1}(a) = H_t(a) - \alpha(R_t - \overline{R}_t)\pi_t(a), \ \forall\ a \neq A_t$$

$H_t(a)$ preference value for action a
α : Learning rate
R_t : Reward received at time *t*.
R_t bar : Average reward received up to and including time *t*.

$\pi_t(a_t)$: The probability [0,1] of choosing action a at time t, as obtained by passing our preference values H through a softmax function.

DESIGN THINKING

K-ARMED BANDIT IMPLEMENTATION

AGENT

Q: action-value estimates

N: Number of times actions chosen in current run

POLICY

Contains method for choosing actions

Contains relevant hyperparameters (e.g., epsilon).

ENVIRONMENT

K : The number of actions.

Q_* : The true expected reward of each arm.

Method for accepting action choice and emitting the reward

Method for resetting reward.

TEST RUNNER

Number of timesteps

Performance histories for reward and optimal action choices (yes/no).

Visualization

Must be able to hold (and compare) multiple agents.

Holds ONE environment.

DESIGN THINKING

K-ARMED BANDIT IMPLEMENTATION

An Agent is assigned to ONE environment at a time (it can interact with one bandit problem at once).

- Agents can learn one problem, then “reset” and learn another.

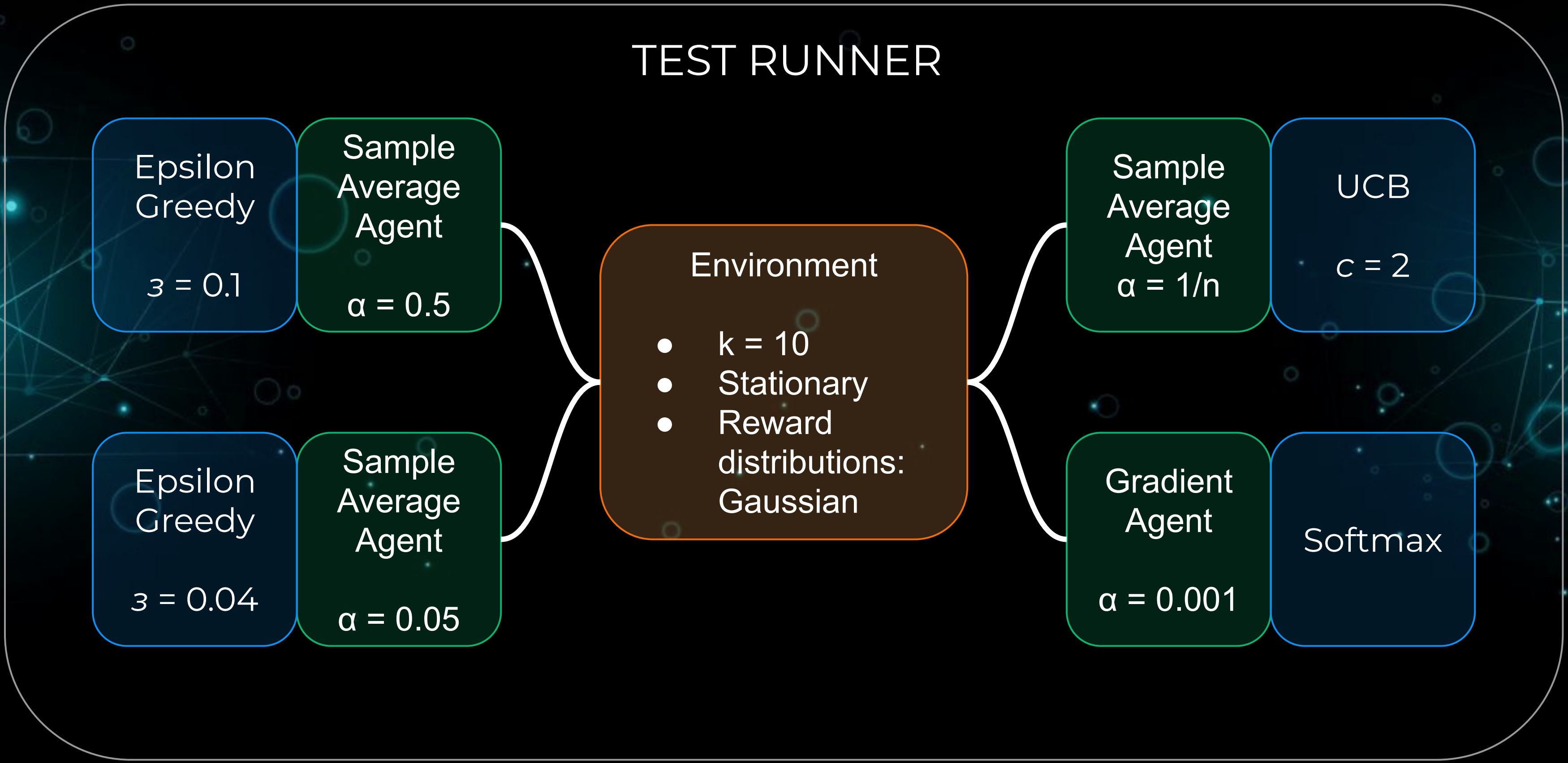
An Agent has ONE policy.

- Just like Environments Agents can swap out one policy for another.

An Environment is a standalone thing that can be interacted with. It has no notion of Agents or Policies.

A Test Runner can have ONE environment at a time. This is for simplicity; we could have multiple but we have no need of that. Environments can be swapped out.

A Test Runner can have MANY Agents at once. This is so we can compare and contrast the performance of different Agents within a single problem space (Environment.


TEST RUNNER
Epsilon Greedy
ɜ = 0.1
Sample Average Agent
α = 0.5
Epsilon Greedy
ɜ = 0.04
Sample Average Agent
α = 0.05
Environment
● k = 10
● Stationary
● Reward distributions: Gaussian
Sample Average Agent
α = 1/n
UCB
c = 2
Gradient Agent
α = 0.001
Softmax