02: Exploration & Control

> An interestic property about RL is that learning is **active**, that means that the actions of the agent impact not only in the reward but in the data that we'll see next. We can deliberaly pick up observe data points, the agent actively seeks out the information it needs to observe.

Exploration vs Exploitation

- *Exploitation* means that the agent maximise performance based on current knowledge.
- *Exploration* increase knowledge.

> There exists a tradeoff between exploration and explotation. We need to gather information to make the best overall decisions, the best long-term strategy may involve short-term sacrifices.

Multi-Armed Bandits

Definition

A multi-armed bandit is a set of distributions:

$$\{R_a | a \in A\}$$

Where

- A is a (known) set of actions
- R_a is a distribution on rewards given action a

Observation

The *action value* for an action a is given by;

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

The *optimal value* is given by;

$$v^* = \max_a \mathbb{E}[R_t | A_t = a]$$

Definition

The *regret* of an action a Is:

$$\Delta_a = v^* - q(a)$$

Observation

We want to minimise the total regret (equivalent to maximising the cumulative reward)

$$L_t = \sum_{n=1}^{t} \Delta_{A_n}$$

Algorithms

We'll discuss several algorithms:

- Greedy
- ϵ-greedy
- Policy granding
- UCB
- Thomson sampling

Observation

A simple estimate for the action value $q(a)$ is the average of the sampled rewards:

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

Where

- $\mathcal{I}(\cdot)$ is the indicatior function
- $\sum_{n=1}^{t} \mathcal{I}(A_n = a)$ is the count of action a

Algorithm (greedy)

> Select the action with the highest estimated value.

Policy

The greedy policy is given by:

$$\pi_t(a) = \mathcal{I}\left(A_t = \mathrm{argmax}_a Q_t(a)\right)$$

Algorithm (ϵ–greedy)

> The problem is that the greedy algorithm can get stuck on a suboptimal action forever, that happens because the greedy algorithm only uses explotation but doesn't do an exploration. The epsilon greedy algorithm solves this by adding some random noise to the action selection process.

Policy

The ϵ-greedy policy is given by;

$$\pi_t(a) = \begin{cases} (1-\epsilon) + \epsilon/|A| & \text{if } Q_t(a) = \max_b Q_t(b) \\ \epsilon/|A| & \text{otherwise} \end{cases}$$

The problem with this algorithm is that it continues picking an action even when it knows it is a pretty bad choice.

Algorithm (Policy gradient)

> Can we learn the policy $\pi(a)$ directly instead of learning values?

Policy

The (softmax) policy-gradient policy is given by:

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

Where

- H_t are the *action preferences*, they are learnable policy parameters.

Learning action preferences

We can use gradient ascent to update the policy parameters H_t as follows:

$$H_{t+1} = H_t + \alpha \nabla_H \mathbb{E}[R_t | \pi_{H_t}]$$

By the log-likelihood trick we have that:

$$\nabla_H \mathbb{E}[R_t|\pi_{H_t}] = \mathbb{E}[R_t \nabla_H \log \pi_H(A_t)]$$

So, the update for the policy parameters is given by:

$$H_{t+1} = H_t + \alpha R_t \nabla_H \log \pi_H(A_t)$$

We can use gradient ascent to compute the optimal action preferences H_t, hence the policy get better over time.

Observation

> Since we use sampled rewards this algorithm does not need value estimates (it is a value free algorithm)

Observation

With the softmax policy the update is given by:

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