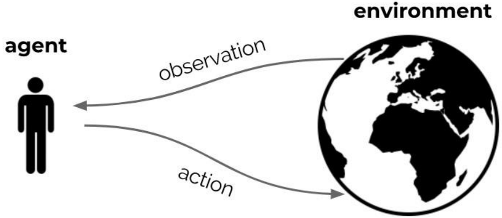
Reinforcement Learning & Intelligent Agents

Lecture 2: Exploration and Exploitation

S. M. Ahsan Kazmi

Recap



- Reinforcement learning is the science of learning to make decisions
- Agents can learn a **policy**, **value function** and/or a **model**
- The general problem involves considering **time** and **consequences**
- Decisions affect the **reward**, the **agent state**, and the **environment state**
- Learning is **active**: decisions impact data

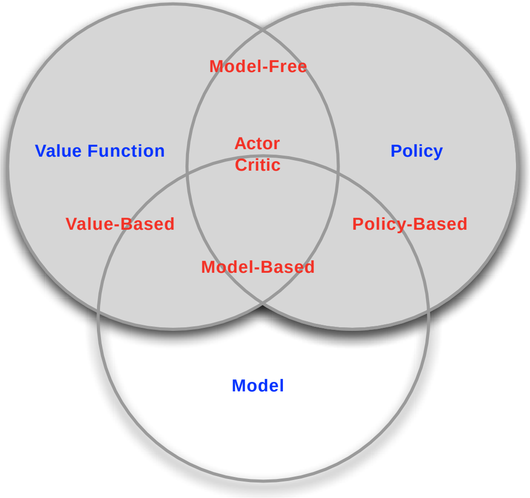
This Lecture

- In this lecture, we simplify the setting
- The environment is assumed to have only a **single state**
  - actions no longer have long-term consequences on the environment
  - actions still do impact **immediate reward,** other observations can be ignored

Agent Categories

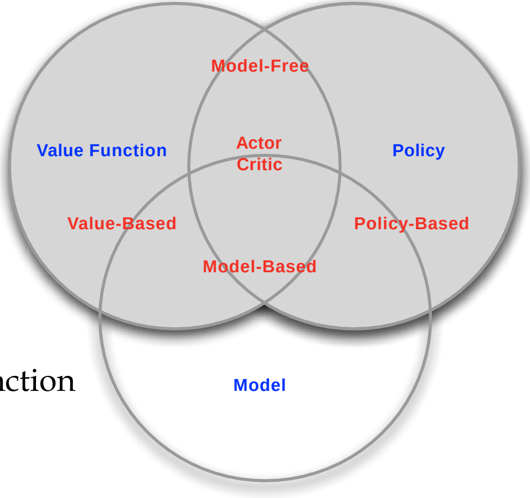
Agent Categories

- Value Based
  - No Policy (Implicit)
  - Value Function
- Policy Based
  - Policy
  - No Value Function
- Actor Critic
  - Policy
  - Value Function



Agent Categories

- Model Free
  - Policy and/or Value Function
  - No Model
- Model Based
  - Optionally Policy and/or Value Function
  - Model



Subproblems of the RL Problem

Learning and Planning

Two fundamental problems in reinforcement learning

- Learning:
  - The environment is initially unknown
  - The agent interacts with the environment
  - The agent improves its policy

- Planning:
  - A model of the environment is given (or learned)
  - The agent plans in this model (without external interaction)
  - a.k.a. reasoning, pondering, thought, search, planning

Prediction and Control

- Prediction: evaluate the future (for a given policy)
- Control: optimise the future (find the best policy)
- These can be strongly related:

$$\pi_*(s) = \underset{\pi}{\operatorname{argmax}}\ v_\pi(s)$$

If we could predict everything, do we need anything else?

Exploration and Exploitation (1)

- Reinforcement learning is like trial-and-error learning
- The agent should discover a good policy
- From its experiences of the environment
- Without losing too much reward along the way
- ***Exploration*** finds more information about the environment
- ***Exploitation*** exploits known information to maximize reward
    - It is usually important to explore as well as exploit

Exploration and Exploitation (2)

- Restaurant Selection
  - Exploitation Go to your favourite 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

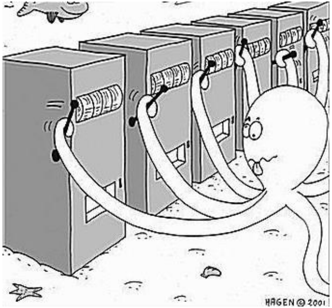
Exploration vs. Exploitation

- Learning agents need to trade off two things
  - **Exploitation**: Maximise performance based on current knowledge
  - **Exploration**: Increase knowledge
- We need to gather information to make the best overall decisions
- The best long-term strategy may involve short-term sacrifices

Formalising the problem

The Multi-Armed Bandit

- A multi-armed bandit is a set of distributions $\{\mathcal{R}_a | a \in \mathcal{A}\}$
- $\mathcal{A}$ is a (known) set of actions (or "arms")
- $\mathcal{R}_a$ is a distribution of rewards, given action a
- At each step t the agent selects an action $A_t \in \mathcal{A}$
- The environment generates a reward $R_t \sim \mathcal{R}_{A_t}$
- The goal is to maximize cumulative reward $\sum_{i=1}^{t} R_i$
- We do this by learning a **policy**: a distribution on $\mathcal{A}$



Values and Regret

The **action value** for action a is the expected reward

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

The **optimal value** is

$$v_* = \max_{a \in \mathcal{A}} q(a) = \max_a \mathbb{E}\left[R_t \mid A_t = a\right]$$

Regret of an action a is

$$\Delta_a = v_* - q(a)$$

The regret for the optimal action is zero

Regret

- We want to minimise **total regret**:

$$L_t = \sum_{n=1}^{t} v_* - q(A_n) = \sum_{n=1}^{t} \Delta_{A_n}$$

- Maximise cumulative reward $\equiv$ minimise total regret
- The summation spans over the full 'lifetime of learning'

Algorithms

Algorithms

- We will discuss several algorithms:
  - Greedy
  - ε-greedy
  - UCB
  - Thompson sampling
- The first three all use **action value estimates** $Q_t(a) \approx q(a)$

Action values

The **action value** for action a is the expected reward

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

A simple estimate is the average of the sampled rewards:

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

$\mathcal{I}(\cdot)$ is the **indicator** function: $\mathcal{I}(\text{True}) = 1$ and $\mathcal{I}(\text{False}) = 0$

The **count** for action a is

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

Action values

This can also be updated incrementally:

$$Q_t(A_t) = Q_{t-1}(A_t) + \alpha_t \underbrace{\left(R_t - Q_{t-1}(A_t)\right)}_{\text{error}},$$

$$\forall a \neq A_t \ : Q_t(a) = Q_{t-1}(a)$$

with

$$\alpha_t = \frac{1}{N_t(A_t)} \qquad \text{and} \qquad N_t(A_t) = N_{t-1}(A_t) + 1,$$

where $N_0(a) = 0$.

- We will later consider other **step sizes** α
- For instance, constant α would lead to **tracking**, rather than averaging

Algorithms: greedy

The greedy policy

One of the simplest policies is **greedy**:

- Select action with highest value: $A_t = \operatorname*{argmax}_a Q_t(a)$
- Equivalently: $\pi_t(a) = \mathbb{I}\left(A_t = \operatorname*{argmax}_a Q_t(a)\right)$ (assuming no ties are possible)

Algorithms: ε-greedy

ε-Greedy Algorithm

- Greedy can get stuck on a suboptimal action forever
  - linear expected total regret
- The **ε-greedy** algorithm:

With probability $1-\varepsilon$ select greedy action: $a = \underset{a \in A}{\operatorname{argmax}}\, Q_t(a)$

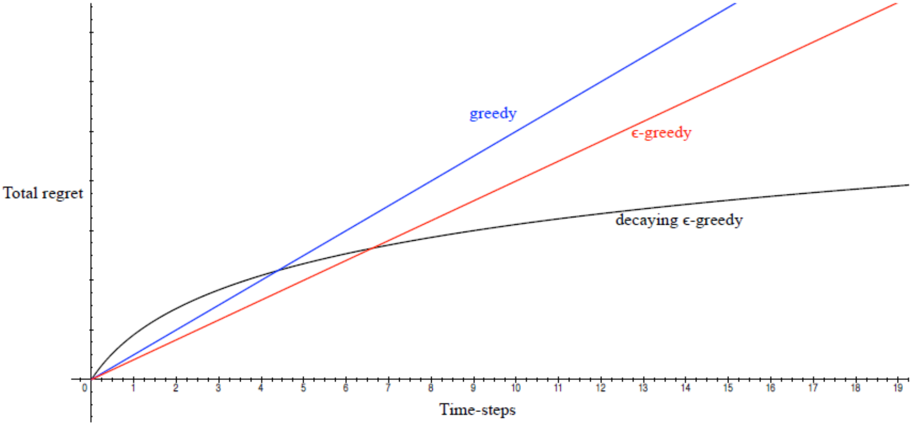
With probability, ε select a random action
Equivalently:

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

- ε-greedy continue to explore
  - ε-greedy with constant ε has linear expected total regret

Decaying ε_t-Greedy Algorithm

- Pick a decay schedule for $\varepsilon_1, \varepsilon_2, \ldots$
- Decaying ε_t -greedy has logarithmic asymptotic total regret!
- Unfortunately, the schedule requires advanced knowledge of gaps
- Goal: find an algorithm with sublinear regret for any multi-armed bandit (without knowledge of R)



- If an algorithm forever explores it will have linear total regret
- If an algorithm never explores it will have linear total regret
- Is it possible to achieve sublinear total regret?

How well can we do?

Theorem (Lai and Robbins)

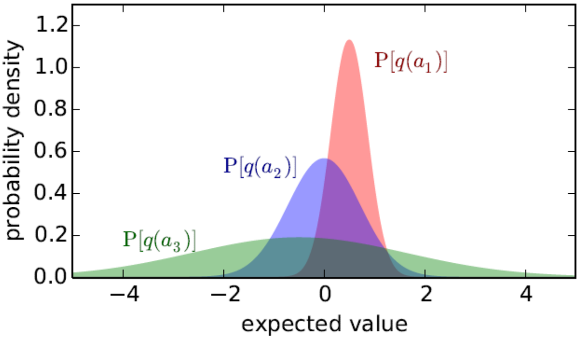
Asymptotic total regret is at least logarithmic in number of steps

$$\lim_{t\to\infty} L_t \geq \log t \sum_{a|\Delta_a>0} \frac{\Delta_a}{KL(\mathcal{R}_a||\mathcal{R}_{a*})}$$

- The performance of any algorithm is determined by similarity between the optimal arm and other arms
- Hard problems have similar-looking arms with different means
- This is described formally by the gap and the similarity in distributions

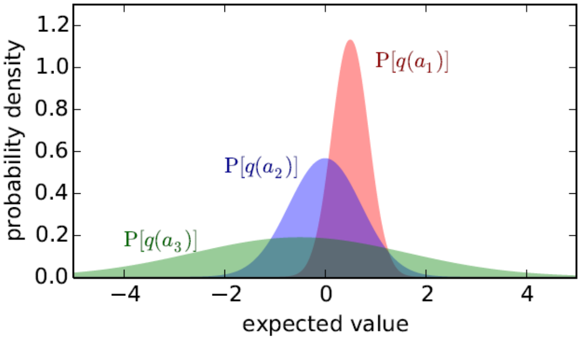
Optimism in the face of uncertainty
Theory: what is possible?

Optimism in the Face of Uncertainty



- Which action should we pick?
- More uncertainty about its value: more important to explore that action
- The more uncertain we are about an action-value, the more important it is to explore that action
- It could turn out to be the best action

Optimism in the Face of Uncertainty



- After picking blue action
- We are less uncertain about the value
- And more likely to pick another action
- Until we home in on the best action

Algorithms: UCB

Upper Confidence Bounds

- Estimate an upper confidence $U_t(a)$ for each action
  such that: $q(a) \leq Q_t(a) + U_t(a)$ with high probability
- Select action maximizing **upper confidence bound** (UCB)

$$a_t = \underset{a \in A}{\operatorname{argmax}}\ Q_t(a) + U_t(a)$$

- The uncertainty should depend on the number of times $N_t(a)$ action a has been selected
  - Small $Nt\ (a) \Rightarrow$ large $Ut\ (a)$ (estimated value is uncertain)
  - Large $N_t(a) \Rightarrow$ small $U_t(a)$ (estimated value is accurate)
- Then a is only selected if either…
  - …$Q_t(a)$ is large (=good action), or
  - …$U_t(a)$ is large (=high uncertainty) (or both)
- Can we **derive** an optimal bound?

UCB

UCB using Hoeffding's Inequality:

$$a_t = \operatorname*{argmax}_{a \in \mathcal{A}} Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}}$$

where c is a hyper-parameter

- Intuition:
  - If Δ_a is large, then $N_t(a)$ is small, because $Q_t(a)$ is likely to be small
  - So either Δ_a is small or $N_t(a)$ is small

Bayesian approaches

Bayesian Bandits

- So far we have made no assumptions about the reward distribution R, except bounds on rewards

- Bayesian bandits exploit prior knowledge of rewards, p [R]

- They compute posterior distribution of rewards p [R | ht]
    - where ht = a1, r1, ..., at−1, rt−1: is the history

- Use posterior to guide exploration:
    - Upper confidence bounds (Bayesian UCB)
    - Probability matching (Thompson sampling)

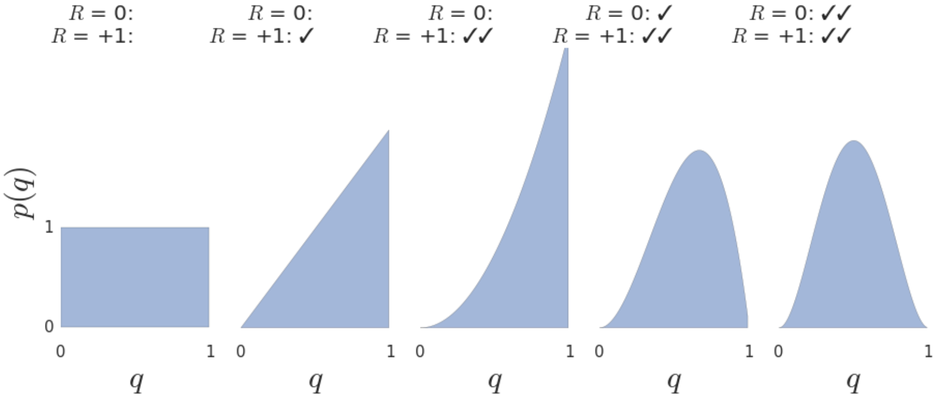
- Better performance if prior knowledge is accurate

Bayesian Bandits: Example

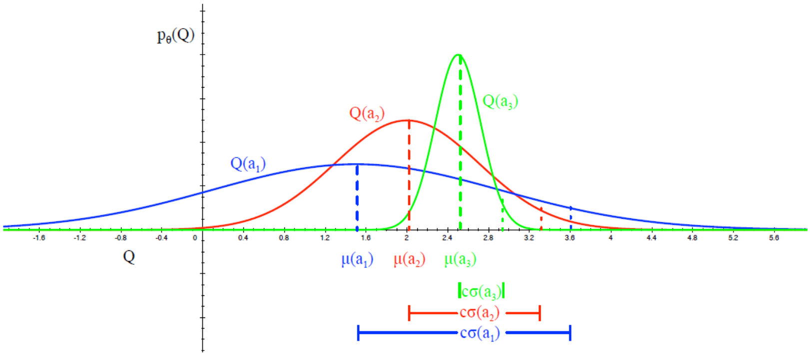
- Consider bandits with **Bernoulli** reward distribution: rewards are 0 or +1
- For each action, the prior could be a **uniform distribution** on $[0, 1]$
- This means we think each value in $[0, 1]$ is equally likely
- Updating the posterior

Bayesian Bandits: Example

Suppose: $R_1 = +1$, $R_2 = +1$, $R_3 = 0$, $R_4 = 0$



Bayesian Bandits with Upper Confidence Bounds



- We can estimate upper confidences from the posterior
  - e.g., $U_t(a) = c\sigma_t(a)$ where $\sigma(a)$ is std dev of $p_t(q(a))$
- Then, pick an action that maximises $Q_t(a) + c\sigma(a)$

Algorithms: Thompson sampling

Probability Matching

- A different option is to use **probability matching**:
- Select action a according to the probability (belief) that a is optimal

$$\pi_t(a) = p\left(q(a) = \max_{a'} q(a') \mid \mathcal{H}_{t-1}\right)$$

- Probability matching is optimistic in the face of uncertainty:
  - Actions have higher probability when either the estimated value is high, or the uncertainty is high
- Can be difficult to compute $\pi(a)$ analytically from posterior (but can be done numerically)

Thompson Sampling

Thompson sampling (Thompson 1933):

- Start with prior beliefs (probability distributions) for each action. Typically, a Beta distribution is used
- Sample $Q_t(a) \sim p_t(q(a)), \forall a$
- Select action maximising sample, $A_t = \text{argmax } Q_t(a)$
- Observe the reward for the chosen action.
- Update the probability distribution for the chosen action based on the observed reward using Bayesian inference.
- Repeat the process for the next time step.

- The idea behind Thompson Sampling is to balance exploration and exploitation by using uncertainty in the estimated values. It adapts over time as more data is collected, making it a powerful and flexible strategy.

- For Bernoulli bandits, Thompson sampling achieves Lai and Robbins lower bound on regret, and therefore is **optimal**

End of lecture