COMP767: Reinforcement Learning

Michael Noukhovitch

Winter 2018,

Notes written from Doina Precup's lectures.

Contents

1 Introduction

1.1 Definitions

Reinforcement learning is:

agent-oriented learning learning by interacting with an environment

trial and error only given delayed evaluative feedback

science of the mind one which is neither natural science nor applied technology

Framework:

1. agent percieves the **state** of the environment
2. based on the state, it chooses an **action**
3. the action gives the agent a **reward**
4. a **policy** aims to maximize the agent's **long term expected reward**

1.2 Key Factors of RL

- trial and error search
- environment is stochastic
- reward may be delayed
- balancing exploration and exploitation

1.3 Classical Challenges

- reward
- information is sequential
- delayed consequences
- balancing exploration/exploitation
- non-stationarity
- fleeting nature of time and online data

2 Bandit

2.1 Single-Armed Bandit

Simplest RL problem

- pull the lever
- get some reward
- choose the best lever!

2.2 k-armed Bandit

- at every time step t, choose an action A_t from k possibilties
- recieve a reward R_t dependent only on the action taken (i.i.d)
- $q_*(a) = \mathbb{E}[R_t | A_t = a], \forall a \in 1, \dots k$

2.2.1 ε-Greedy Action Selection

Solve exploration/exploitation by usually selecting exploit (best action so far), but exploring (random action) with probability ε

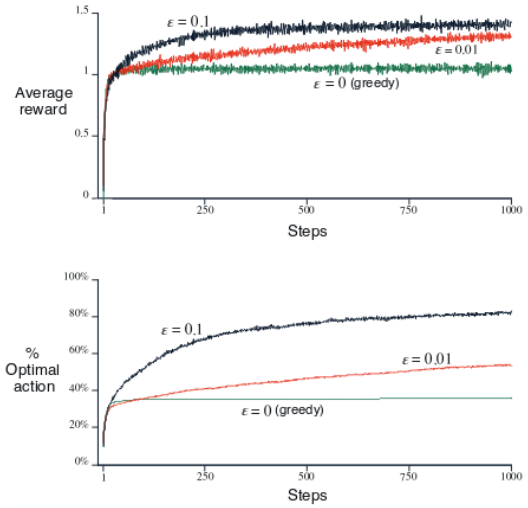

Figure 1: ϵ-greedy methods on 10-arm bandit

2.3 Learning Rules

Learn the best policy by learning the reward for an action

2.3.1 Averaging

For a single action, update the new estimate based on old estimate and step size (α), with all actions being equal

$$Q_{n+1} = Q_n + \alpha(R_n - Q_n)$$

2.3.2 Recency-Weighted Average

non-stationary if the true action values change slowly over time

if so, then we need to put more weight on recent samples (using exponential)

$$Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$$

2.3.3 Optimistic

Previously we assumed $Q_1(a) = 0$, but we can start optimistically (e.g. $Q_1(a) = 5$) to encourage early exploration

2.3.4 Upper Confidence Bound

Reduce exploration over time after starting confident

- estimate upper bound on true action values
- select the action with the largest upper bound

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

2.4 Evaluations

regret the difference between best option and the one we chose $\max_a q_*(a) - q_t(a)$

expected total regret $\mathbb{E}[\sum_t \text{regret}_t]$ (optimal for UCB, Thomson sampling)

best response regret for T experimental trials after policy is fixed

2.5 Gradient-Bandit Algorithms

Don't need to learn specific rewards, just learn the **preference** $H_t(a)$, and try and make the probability of choosing an action $\pi_t(a)$ be proportional to it.

$$\begin{aligned}\pi_t(a) &\propto e^{H_t(a)} \\ &= \frac{e^{H_t(a)}}{\sum_b e^{H_t(b)}}\end{aligned}$$

if the reward for an action is better than average, increase its preference

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

where $\bar{R}_t$ = average R_i

2.6 Conclusions

- simple methods that can be built on
- learn from feedback
- appear to have a goal

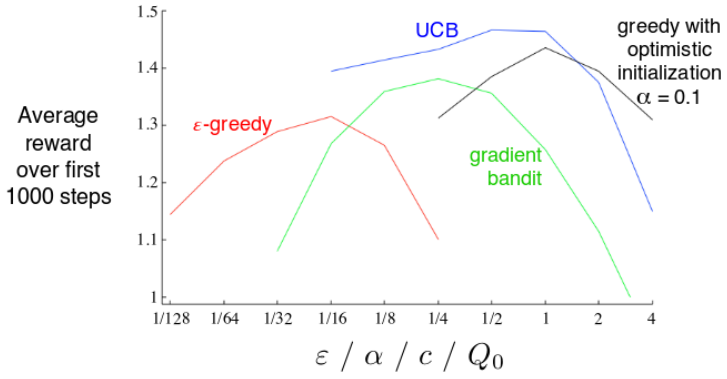

Figure 2: bandit algorithm comparison