1 Preliminaries

In this short section we present the purpose of the field of Reinforcement Learning and introduce the fundamental concepts associated with it. A simple but broad characterization of RL is the following: ***Reinforcement Learning is a set of computational methods designed to automate learning in unknown environments where the emphasis is on making decisions to achieve predefined goals***. The core paradigm is that the system learns through direct interactions with the environment (i.e. there is continual feedback loop between the agent and the environment) and does not necessarily require any outside supervision. Since the only signal the agent receives comes from the interactions with the environment, there is constant conflict between two aspects of these interactions: ***exploration*** – trying different actions to see the outcome and ***exploitation*** – using the actions that yielded the best results in the past. The fundamental concepts of reinforcement learning are:

- ***agent*** – an entity that can: ***observe*** (at least partially) the environment, ***perform actions*** that change itself and the environment and ***receive rewards*** in connection with the interactions with the environment
- ***environment*** – the reality outside of an agent
- ***policy / control*** – defining the behavior of the agent

2 Multi-armed Bandits

Our first example of an RL system will be the well-known problem of k-armed bandits. In this problem we have a finite set of k possible actions $\mathcal{A}$ and a conditional probability distribution of the rewards r given the chosen action a i.e. $p(r \mid a)$. This probability distribution is obviously not known a priori and thus our goal is to find a policy which maximizes our expected reward over some finite number of steps T. Note that if we knew the probability distributions ahead of time then in each step we should just choose the action a^* such that

$$a^* = \arg\max_{a \in \mathcal{A}} \mathbb{E}_{r \sim p(r|a)}[r]$$

since

$$\mathbb{E}_{r_1 \sim p(r|a_1), \ldots, r_T \sim p(r|a_T)} \left[\sum_{t=1}^{T} r_t \right] = \sum_{t=1}^{T} \mathbb{E}_{r_t \sim p(r|a_t)}[r_t] \,.$$

2.1 Value function methods

One of two classes of methods used in the k-armed bandits problem are based on the so called ***value function*** from the set of possible actions A to the set of real numbers, which at step t is given by

$$q_t(a) = \frac{1}{n_t(a)} \sum_{i=1}^{t-1} r_i \delta_{a_i,a} \,, \quad n_t(a) = \sum_{i=1}^{t-1} \delta_{a_i,a} \,,$$

where $a_i \in \mathcal{A}$ is the action chosen at step i, $n_t(a)$ is the number of times action a was chosen up until timestep t and δ denotes the Kronecker's delta. Let's note that this expression satisfies the following recursive equation

$$\boxed{\begin{aligned} n_{t+1}(a) &= n_t(a) + \delta_{a_t,a} \\ q_{t+1}(a) &= q_t(a) + \frac{\delta_{a_t,a}}{n_{t+1}(a)} \big[r_t - q_t(a) \big] \end{aligned}}$$

Based on the definition of value function and the recursive equations for the update we can devise several methods of choosing the action at given step which differ by the way they balance exploration and exploitation.

2.1.1 ϵ-Greedy

Given the value function at step t, the simplest possible choice of action is the greedy one i.e.

$$a_t = \arg\max_{a \in \mathcal{A}} q_t(a) \,.$$

This approach is pure exploitation an no exploration and thus we predict that it won't perform well. A simple modification of this method which allows for any exploration is the ϵ-greedy choice where with probability ϵ we randomly choose any action from $\mathcal{A}$ and with probability $1 - \epsilon$ greedily choose the one with maximal value function. The value of ϵ is chosen beforehand and can have a big impact on the overall performance of this algorithm. Of course if $\epsilon = 0$ then we just go back to the greedy choice of action.

Two possible modifications of this simple algorithm can make it perform better. First of all if we look at the equation for the value function update we can generalize it a bit and write it as

$$q_{t+1}(a) = q_t(a) + \delta_{a_t,a} \alpha_t(a) \big[r_t(a) - q_t(a) \big] \,,$$

where $\alpha_t(a)$ is called step-size. We can thus choose step-size beforehand and update the value function according to this equation instead of the one before. Additionally we can use the so called ***optimistic initialization*** i.e. choose suitable values of $q_0(a)$ different than 0 (which is on the contrary called the ***realistic initialization***). This makes the algorithm prefer exploration in the initial phase.