Reinforcement Learning(1a/3)

Bruno Bouzy

23 septembre 2014

This document consists in the « Reinforcement Learning » chapter of the « Agent oriented learning » teaching unit of the Master MI computer course. It is based on parts I and II of (Sutton & Barto 1998). The figures contained in this document are directly taken from the html version of (Sutton & Barto 1998) on the web.

Introduction

Generalities

Reinforcement Learning (RL) consists in learning what to do, how to associate actions to situations in order to maximize future rewards. The learner is not told which action to perform, but he must uncover which actions result in the largest rewards by trying the actions. In the most interesting case the actions may effect on long term rewards. These two properties – trial-and-error searching and long term reward – are the main two features of RL. RL does not need an oracle, and is not part of supervised learning. Instead the agent interact with its environment that gives him the feedback. In RL, the challenge is the balance between exploitation (how to act optimally given the current knowledge) and exploration (which actions to choose to gain knowledge).

Basic elements of RL

In addition to the agent and its environment, some elements can be distinguished: the policy, the reward function, the transition function and the value functions. The policy belongs to the agent, and enables him to choose its action. The reward function belongs to the environment and informs the agent of its reward. The transition function is part of the environment that outputs the next state of the agent according to the previous one and its action. The value functions yields what is good on the long term, and belongs to the learning agent.

Learning action values with feedback

An important feature of RL is evaluating the actions on the basis of the feedback received by executing these actions, and not to use an advice that could be given by an oracle, as done in supervised learning. This obliges the agent to explore all actions to find out their consequences.

The multi-armed bandit (MAB) problem, the greedy method

Let us assume the following. We face a choice between N actions. After choosing an action, we will receive a return – a number – according to a probability distribution of the returns of this action. The probability distribution maps a return with a probability density. The

probability distributions of the actions are unknown. We are allowed to repeat this process X times. Our aim is to maximize the cumulated return obtained after X times. We are concerned with problems with X large, or even X going to infinity. In this problem, the agent remains in the same state.

In the multi-armed bandit formulation of this problem, a bandit faces a slot machine with N arms. The bandit has to choose an arm, pull it, and get the reward. The bandit aims at maximizing the cumulated reward over time.

The important point is that the rewards received are non deterministic. If you pull the same arm twice, you are not garanteed to get the same reward.

Each action a has a theoretical mean corresponding to its probability distribution. Let us call it m_a* or $Q^*(a)$. If the probability distributions were known, the optimal behaviour would be to pull the arm with the largest theoretical mean.

Since the theoretical means are unkown, we have to estimate them. Let us call $m^\wedge_a$ or $Q_t(a)$ the empirical mean of action a. t is the time at which the action is estimated. The empirical mean is the mean of the actual returns observed until now. The empirical mean is our estimation of the theoretical mean. At time t, action a has been selected k_a times yielding rewards r_1, r_2, …, r_{ka}, we have :

$$Q_t(a) = (r_1 + r_2 + \ldots + r_{ka})/k_a \qquad (1)$$

If ka=0, we define $Q_0(a) = 0$. As ka -> ∞, $Q_t(a)$ -> $Q^*(a)$.

The greedy method selects the action with the highest estimation.

$$a_{greedy} = \text{argmax}_{a \text{ in } A} \; m^\wedge_a \qquad (2)$$

a_{greedy} is called the greedy action. There can be several greedy actions. We will say that when we are greedy, we *exploit* the current knowledge of the actions. We will say that when we choose an action which is not the best one according to the greedy method, we *explore* to refine our current knowledge of the actions. At each timestep, the question we face is to choose between exploration and exploitation. This is the exploration/exploitation dilemma.

The ε-greedy method

With probability ε, the ε-greedy method selects an action at random, and with probability 1-ε, the ε-greedy method selects a greedy action. With the ε-method, all actions are selected infinitely. Therefore the estimations converge to their theoretical means. The probability to select the best action approaches 1-ε.

To show the efficiency of these methods, we have a test set of 2000 MAB problems with 10 actions. For each action a, the rewards were selected with a gaussian distribution with mean $Q^*(a)$ and variance 1.

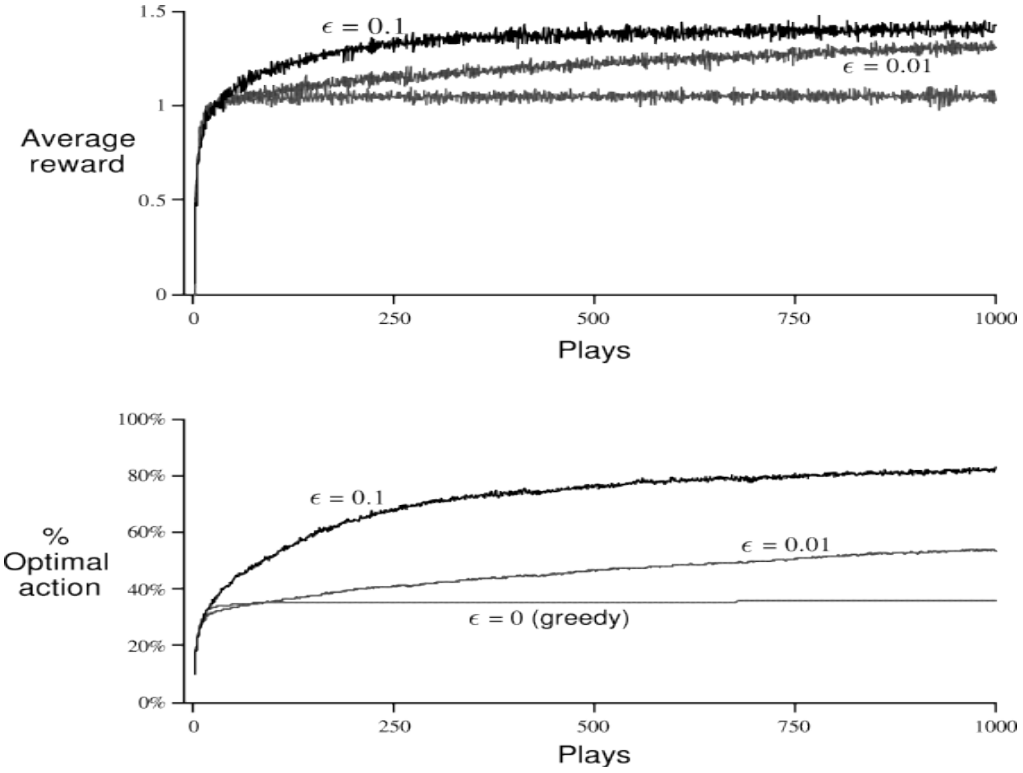

Figure 1 : Mean rewards and percentages the optimal action has been selected.

Figure 1 compare the greedy method to two ε-greedy methods ($\varepsilon = 0.1$ and $\varepsilon = 0.01$). At the beginning the greedy method improves slightly faster than the ε-greedy methods. But then, it is stuck at a level inferior to the levels of the ε-greedy methods. The curve at the bottom shows that the greedy method selects the optimal action one third of the times. The ε-greedy methods continue to explore. With $\varepsilon = 0.1$, it explores more but gathers 91% of the optimal reward only. With $\varepsilon = 0.01$, the method improves more slowly but gathers 99% of the optimal reward.

The softmax method

Although very popular in RL, when exploring, the ε-greedy methods selects its action at random. To improve exploration, the softmax method prefers to explore actions that are near-optimal before exploring bad actions. Such a simple selection method is to choose action a with the probability:

$$Q_t(a)/\sum_b Q_t(b) \qquad (2)$$

The softmax method is an improvemnt of the previous one by using the following probability :

$$\exp(Q_t(a)/\tau)/\sum_b \exp(Q_t(b)/\tau) \qquad (3)$$

τ is named the temperature. With a high temperature, the softmax method looks like the random selection with uniform probabilty, and exploration is underlined. With a small temperature, it looks like the greedy method because the probability of non-optimal action is

negligible in front of the probability of the best action that is nearly 1, and exploitation is underlined.

The UCB method

The UCB method selects its arm b_{ucb} with the follwing formula:

$$b_{ucb} = \text{argmax}_{\ b \text{ in } B}\ \ m\hat{}(b) + C \text{ sqrt } (\log(T)/N(b)) \qquad (4)$$

m^(b) is the empirical mean of arm b. T is the total number of trials over all the arms. N(b) is the number of trials of arm b. The first term of (4) corresponds to exploiting: using the empirical mean value. The second term of (4) corresponds to exploring. The lower N(b) the greater exploration. At a time t, if an arm b is not explored enough, its exploration term is high, and b is explored. If b is good, its mean value increases and b is selected again. As long as b is selected, its exploration term is decreasing. Meanwhile, its empirical mean is converging, and the exploration term of all other arms are increasing - with the log(T) term. Therefore at some time, an arm different from b while be selected. An arm cannot stay unexplored. All arms will be visited an unfinitely and the empirical means converge to the theoretical means. UCB means « Upper Confidence Bound ».

Evaluation vs Instruction ; Reinforcement vs Supervised learning.

In the MAB problem, the reward is quantitative. The reward evaluates the action. This is very different from Supervised Learning (SL) in which the return informs the agent about the correctness of the action or its uncorrectness. To feel the difference between RL and SL, let us assume we have 100 actions. We select action 32 and, in the RL approach, the environment tells us that this action is worth 7.2. In the SL approach, an oracle tells us that action 32 is correct.

Let us consider the MAB problem with 2 actions 1 and 2. The return can be « success » or « failure » (0 or 1). Let us assume each action has a probability of success, p1 and p2. In the SL approach, the agent memorize +1 for action 1 (resp. 2) when 1 (resp. 2) was selected and successful, or when 2 (resp. 1) was selected but was a failure. [Sutton & Barto] shows figure 2 and says that they are difficult problems in which the SL method cannot work, and easy problems in which SL works. Meanwhile, RL works for all problems: easy or difficult. Could you give an insight about this interpretation ?

Incrémentality

RL methods estimate action values with mean value. The following formula enables a method to update the mean value without memorizing past reward values:

$$Q_{k+1} = Q_k + (r_{k+1} - Q_k)/(k+1) \qquad (5)$$

Generally, RL uses the following kind of updating:

$$\text{newEstim} = \text{oldEstim} + \text{step} \,.\, (\text{target} - \text{oldEstim}) \qquad (6)$$

(target – oldEstim) is an estimation error. step is the learning step, often named α.