Hands on Reinforcement Learning*

*

Deepasha
Computer Science and Engineering
IIT Bombay

Topic 1: Multi-Arm Bandits

I. Introduction

Consider the following learning problem. You are faced repeatedly with a choice among n dfferent options, or actions. After each choice you receive a numerical reward chosen from a stationary probability distribution that depends on the action you selected. Your objective is to maximize the expected total reward over some time period,

II. Description

This is the original form of the n-armed bandit problem, so named by analogy to a slot machine, or one-armed bandit," except that it has n levers instead of one. Each action selection is like a play of one of the slot machine's levers, and the rewards are the payoffs for hitting the jackpot. Through repeated action selections you are to maximize your winnings by concentrating your actions on the best levers.

If you maintain estimates of the action values, then at any time step there is at least one action whose estimated value is greatest. We call this a greedy action. If you select a greedy action, we say that you are **exploiting** your current knowledge of the values of the actions. If instead you select one of the nongreedy actions, then we say you are **exploring**, because this enables you to improve your estimate of the nongreedy action's value.

III. Action-Value Methods

We denote the true (actual) value of action a as q(a), and the estimated value on the t_{th} time step as $Q_t(a)$. The true value of an action is the mean reward received when that action is selected. One natural way to estimate this is by averaging the rewards actually received when the action was selected. In other words, if by the t_{th} time step action a has been chosen $N_t(a)$ times prior to t, yielding rewards $R_1, R_2, .., R_{N_t(a)}$, then its value is estimated to be

$$Q_t(a) = \frac{R_1 + R_2 + .. + R_{N_t(a)}}{N_t(a)}$$

If $N_t(a)$ = 0, then we define $Q_t(a)$ instead as some default value, such as $Q_1(a)$ = 0. By the law of large numbers, $Q_t(a)$ converges to q(a). We call this the sample-average method for estimating action values.

A. Greedy method

The simplest action selection rule is to select the action (or one of the actions) with highest estimated action value, that is, to select at step t one of the greedy actions, A_t^*, for which $Q_t(A_t^*) = max_a Q_t(a)$. This greedy action selection method can be written as Greedy action selection always exploits current knowledge to maximize immediate reward;

B. e-Greedy method

A simple alternative is to behave greedily most of the time, but every once in a while, say with small probability ϵ, instead to select randomly from amongst all the actions with equal probability independently of the action value estimates.

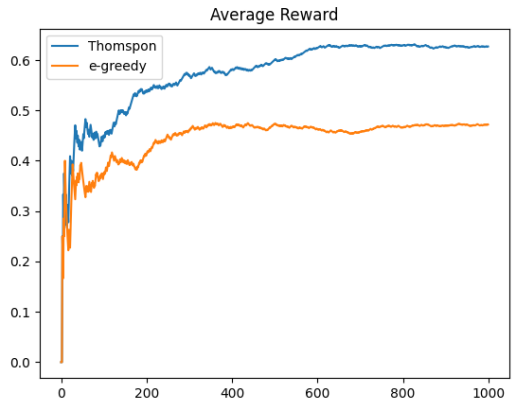

Fig. 1. Reward function of e greedy

C. Problems with Action-Value Methods

Memory and computational requirements grow over time without bound. That is, each additional reward following a selection of action a requires more memory to store it and results in more computation being required to determine $Q_t(a)$.

IV. Upper-Confidence-Bound Action Selection

The idea of this upper confidence bound (UCB) action selection is that the square-root term is a measure of the uncertainty or variance in the estimate of a's value. The quantity being max'ed over is thus a sort of upper bound

on the possible true value of action a, with the c parameter determining the confidence level. Each time a is selected the uncertainty is presumably reduced; $N_t(a)$ is incremented and, as it appears in the denominator of the uncertainty term, the term is decreased. On the other hand, each time an action other a is selected t is increased; as it appears in the numerator the uncertainty estimate is increased. The use of the natural logarithm means that the increase gets smaller over time, but is unbounded; all actions will eventually be selected, but as time goes by it will be a longer wait, and thus a lower selection frequency, for actions with a lower value estimate or that have already been selected more times.

$$A_t = argmax_a[Q_t(a) + c\sqrt{\frac{lnt}{N_t(a)}}]$$

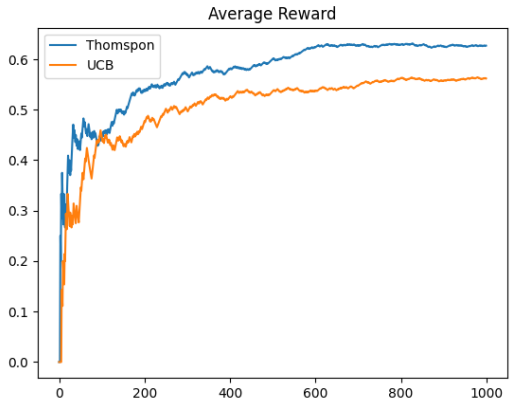

Fig. 2. Reward function of UCB.

Topic 2: Finite Markov Decision Processes

V. Definitions

An MDP M = (S,A, p, R, γ) has these elements.

- S: a set of states. Let us assume S = $s_1, s_2, ..., s_n$, and hence $\mathbf{card}(S)$= n.
- A: a set of actions. Let us assume A = $a_1, a_2, ..., a_k$, and hence $\mathbf{card}(A)$ = k.
- p: a transition function. p(s,a,) is a probability distribution over S.
- R: a reward function.For s, $s_0 \in$ S, a $\in$ A: R(s, a, s_0) is the (numeric) reward for reaching s_0 by starting at s and taking action a.
- The discount rate determines the present value of future rewards
- Value function slides.

VI. The Agent-Environment Interface

The reinforcement learning problem is meant to be a straightforward framing of the problem of learning from interaction to achieve a goal. The learner and decision-maker is called the **agent**. The thing it interacts with, comprising everything outside the agent, is called the **environment**. These interact continually, the agent selecting actions and the environment responding to those actions and presenting new situations to the agent. The environment also gives rise to rewards, special numerical values that the agent tries to maximize over time. A complete specification of an environment defines a task, one instance of the reinforcement learning problem.

More specifically, the agent and environment interact at each of a sequence of discrete time steps. At each time step t, the agent receives some representation of the environment's state, $S_t \in$ S, where S is the set of possible states, and on that basis selects an action, $A_t \in A(S_t)$ where $A(S_t)$ is the set of actions available in state S_t. One time step later, in part as a consequence of its action, the agent receives a numerical reward, R_{t+1} and finds itself in a new state, S_{t+1}

At each time step, the agent implements a mapping from states to probabilities of selecting each possible action. This mapping is called the agent's policy and is denoted π_t, where $\pi_t(a|s)$ is the probability that A_t = a if S_t = s. Reinforcement learning methods specify how the agent changes its policy as a result of its experience. The agent's goal, roughly speaking, is to *maximize the total amount of reward* it receives over the long run.

VII. MDP Planning

Given M = (S,A, p, R, γ), find a policy π

- Every MDP is guaranteed to have a deterministic, Markovian, stationary optimal policy.
- An MDP can have more than one optimal policy.
- However, the value function of every optimal policy is the same, unique "optimal value function" V

VIII. Policy Evaluation

Policy evaluation refers to the (typically) iterative computation of the value functions for a given policy. Guaranteed to have a unique solution if gamma $\leq$ 1.
Consider a sequence of approximate value functions $V_0; V_1; V_2; ...$, each mapping S to R. The initial approximation, V_0, is chosen arbitrarily, and each successive approximation is obtained by using the Bellman equation for V^π as an update rule:

$$V_{k+1} = \sum_a \pi(a|s) \sum_{s',a} p(s', r|s, a)(r + \gamma V_k(s'))$$

IX. Policy Improvement

A. Theorem

Given π, pick one or more improvable states, and in them, switch to an arbitrary improving action. Let the resulting policy be π '
For $\pi \in \Pi$ and $s \in S$,

$$IA(\pi, s) = a \in A : Q^\pi(s, a) > V^\pi(s)$$

For $\pi \in \Pi$,

$$IS(\pi) = s \in S : |IA(\pi, s)| > 1$$

Initialize V arbitrarily, e.g.,$V(s) = 0$, for all $s \in \mathcal{S}^+$

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
  $\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

Fig. 3. Pseudo code of Value Iteration.

B. Implication

(1) If IS(π) = ϕ, then π is optimal, else
(2) If π' is obtained by policy improvement then π' dominates π
(3) V^{π^*} satisfies the Bellman optimality.

C. Algorithms

- Howard's Policy Iteration: Greedy; switch to all improvable states
- Random Policy Iteration: Switch a non-empty subset of improvable states chosen uniformly at random.
- Simple Policy Iteration: Assume a fixed indexing of states and then switch the improvable state with the highest index.

1. Initialization
  $V(s) \in \Re$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
    Repeat
      $\Delta \leftarrow 0$
      For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_{s'} \mathcal{P}^{\pi(s)}_{ss'} \left[\mathcal{R}^{\pi(s)}_{ss'} + \gamma V(s')\right]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
    until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
    policy-stable $\leftarrow true$
    For each $s \in \mathcal{S}$:
      $b \leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
      If $b \neq \pi(s)$, then *policy-stable* $\leftarrow false$
    If *policy-stable*, then stop; else go to 2

Fig. 4. Pseudo code of Policy Iteration.

Topic 3: Reinforcement Learning

X. The Control Problem

- A learning algorithm L is a mapping from the set of all histories to the set of all (probability distributions over) arms.
- Actions are selected by the learning algorithm (agent); next states and rewards by the MDP (environment).
- For t $\geq$ 0, let h^t = $(s^0, a^0, r^0, s^1, a^1, r^1, ..)$ denote a t-length history.
- Can we construct L such that:

$$lim_{T>\infty}[\sum_{t=0}^{T-1} P(a_t = L(h_t) is an optimal action for s_t)]/T = 1$$

XI. The Prediction Problem

- We are given a policy π that the agent follows. The aim is to estimate V^π
- For t $\geq$ 0, let h^t = $(s^0, a^0, r^0, s^1, a^1, r^1, ..)$ denote a t-length history.
- A learning algorithm L is a mapping from the set of all histories to the set of all (probability distributions over) arms.
- In other words, at each step t the learning algorithm provides an estimate V'_t

$$lim_{t>\infty} V'_t = V_\pi$$

Acknowledgment

Would like to thank both my mentors for giving me an opportunity to to explore the field of reinforcement learning.

References

- Reinforcement Learning: An Introduction. Second edition, in progress. Richard S. Sutton and Andrew G. Barto c 2014, 2015. A Bradford Book. The MIT Press.
- CS747- Foundations of Intelligent and Learning Agents