1: Introduction to RL

Ameya Godbole

1 Acknowledgement

A thanks to:

1. the course instructor: Prof. Balaraman Ravindran (IIT Madras)
2. National Programme on Technology Enhanced Learning (NPTEL), India

2 Roots in behavioural psychology

Pavlov's dog: Dog was contitioned to expect food after a bell. It formed the association resulting in it starting to salivate at the sound of a bell even when no food was given. Based on experiments in this vein, theories were formulated about how learning proceeds.

3 What is reinforcement learning

- Reinforcement learning is a mathematical formulation for a **trial-and-error** kind of learning.
- It is a paradigm where we learn about a system through interaction.
- Learning about stimuli and actions through **rewards and punishment alone**.
- No detailed supervision is available. E.g. no one give a value of force to be applied on the pedal when cycling.
- Rewards and punishments may be delayed (temporarily) with respect to the event that caused it. So an association between cause and effect needs to be learned
- Sequence of actions (may be) needed to earn reward

4 RL Framework

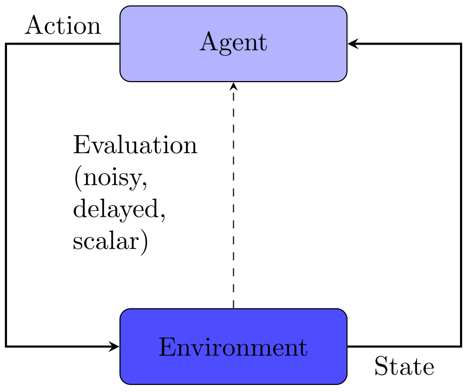

Figure 1: General framework

Crux: Agent work in close interaction with environment. It senses the state of the environment, take action and see how state is affected. Don't choose actions that are beneficial in the current state but actions that put you in a beneficial state in the future

Assumption: Environment gives some form of evaluation. This evaluation may be noisy and temporally delayed. In reality, all sensory input received from the surroundings is processed and some part of the brain interprets some of the sensory input as reward/punishment. Evaluation is assumed (defined) to be a scalar. This scalar is to be maximised over time.

Not supervised learning: In RL, there is very *sparse* supervision and no target output. Since there is no error function given by the environment, trial-and-error is essential for estimating gradient.

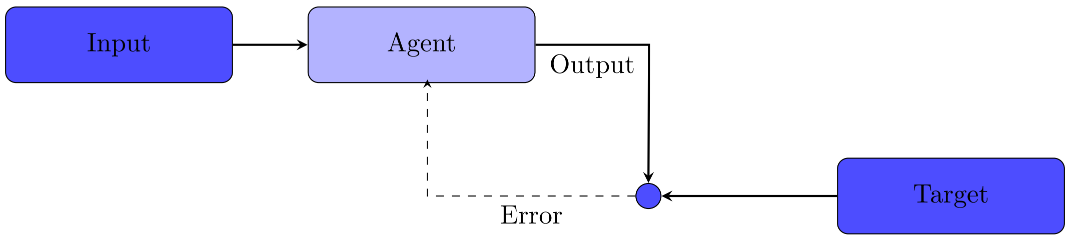

Figure 2: Supervised learning framework

Not unsupervised learning: There is sparse supervision. Also, pattern discovery is not the primary goal in RL.

Temporal difference learning: The prediction of eventual outcome at time *t+1* will be better than the prediction at time *t*. Hence use the later prediction to adjust earlier prediction. TD learning seems to explain some neurotransmitter behaviour in the brain.

Explore-Exploit dilemma: Explore to find other/more profitable actions and exploit to act according to the best actions already made.

5 Immediate Reinforcement Learning Problems

In this category of problems, reward is obtained immediately after an action i.e. there is no time delay between action and reward.

Notation: At time t, an action a_t results in reward R_t.

Notice that we do not mention a time/state dependent input. The agent here receives the same input at all time instants, i.e. it always plays the same one-step game.

The outcome may be stochastic. The same action can result in different rewards at different instances. Thus, the effect of different actions should be *explored*. However, we are interested in eventually *exploiting* the knowledge acquired.

The payoff is assumed to be obtained from a distribution that depends on the action chosen but does not change over multiple experiments[1]. If the distributions were known, the optimal move would be to select the action with highest expected payoff. Thus, the problem here is to figure out the different distributions.

Notation: Associated with each action a let the **true** expected payoff be $q_*(a)$

5.1 Bandit problems

Notation: *One-armed bandit* is a slot machine where you put in a coin, pull a lever (arm) and get reward(mostly none hence the name bandit)

[1]if X is RV with cdf C, then distribution can be sampled by returning $C^{-1}(uniform(0,1))$

Multi-arm bandit is essentially this slot machine with multiple levers with different payoff distributions.

Solution of multi-arm bandit problem can involve:

1. Asymptotic correctness: There are not constraints but give a guarantee that eventually the agent selects the arm with the highest expected payoff.
2. Regret optimality[2]: Suppose the agent expores the problem space and accumulates reward till time t. The difference between this reward and the reward obtained if the optimal action was followed from the beginning is the *regret*. Regret optimality aims to maximize the total reward obtained (even in the learning phase), i.e. we want to reach the maximum expected payoff quickly. It is an established result that regret grows/accumulates at least at rate $log(t)$.
3. Probably approximately correct (PAC) optimality: With a high probabilty (probably) the arm returned as result will give expected payoff close to optimal expected payoff (approximately correct). $(\epsilon, \delta) - PAC$ means that with probability $(1-\delta)$ the solution returned will be within ϵ of the optimal solution i.e. $Pr\{q(result) \geq q_*(a^*) - \epsilon\} \geq (1-\delta)$. $(\epsilon, \delta) - PAC$ optimality is smallest number of times the arms are sampled such that the agent can give the $(\epsilon, \delta) - PAC$ gurantee.

1 is a question of correctness, 2 is a question of rate of convergence and 3 is a sample complexity question.

5.2 Action-Value Methods

Notation: $Q_t(a)$ is the *estimated* expected payoff of an action i.e. an approximation of $q_*(a)$ the true expected payoff.

$Q_t(a) = \frac{\sum_{i=1}^{t} 1_{a_i=a} . R_i}{\sum_{i=1}^{t} 1_{a_i=a}}$

1. Greedy policy: action chosen $= \underset{a}{\operatorname{argmax}}\, Q_t(a)$
2. ϵ-greedy: With probability $(1-\epsilon)$ be greedy and with probability ϵ explore (choose any random action)
3. softmax: $Pr\{a_{t+1} = a\} = \frac{e^{Q(a)/\beta}}{\sum_{b=1}^{n} e^{Q(b)/\beta}}$ where β is a temperature parameter. High β spreads probability (more random) over actions while low β (cooler,calmer) gives peakier distribution.

Asymptotic correctness of the ϵ-greedy approach is ensured as exploration and exploitation are balanced. However, to ensure choice of only the optimal arm in the long run, *epsilon* must be cooled down. Decreasing ϵ as 1/t is proposed but this converges to zero only asymptotically, so instead ϵ can be cooled in steps.

5.3 Incremental Implementation

To calculate $Q_t(a)$ without having to store the previous values,

$Q_t(a) = Q_{t-1}(a) + \frac{1_{a_t=a}}{n_a + 1_{a_t=a}}[R_t.1_{a_t=a} - Q_{t-1}(a)]$

This is an unbaised estimate of the mean payoff of action a. This converges if the reward is obtained from a stationary process. For non-stationary reward process, the update rule must be modified to give more weight to recent samples than long-past ones. Modified update rule:

$Q_t(a) = Q_{t-1}(a) + \alpha.1_{a_t=a}[R_t.1_{a_t=a} - Q_{t-1}(a)]$

where α, step-size parameter, lies in (0,1]. When expanded, we see that the weight of reward R_i is $\alpha(1-\alpha)^i$ which is exponentially decaying.

[2] E.g.: we design a sophisticated online algorithm that deals with various issues of uncertainty and decision making, and sell it to a client. Our algorithm runs for some time and incurs a certain loss. We would like to avoid the embarrassment that our client will come back to us and claim that in retrospect we could have incurred a much lower loss if we used his simple alternative policy π. The regret of our online algorithm is the difference between the loss of our algorithm and the loss using π.

Book notes
Sometimes it is convenient to vary the step-size parameter from step to step. Let $\alpha_k(a)$ denote the step-size parameter used to process the reward received after the kth (k $= n_a + 1_{a_t=a}$) selection of action a. As we have noted, the choice $\alpha_k(a) = \frac{1}{k}$ results in the sample-average method, which is guaranteed to converge to the true action values by the law of large numbers. But of course convergence is not guaranteed for all choices of the sequence k (a). A well-known result in stochastic approximation theory gives us the conditions required to assure convergence with probability 1:

$\sum_{k=1}^{\infty} a_k = \infty$ and $\sum_{k=1}^{\infty} a_k^2 < \infty$

The first condition is required to guarantee that the steps are large enough to eventually overcome any initial conditions or random fluctuations. The second condition guarantees that eventually the steps become small enough to assure convergence. Note that both convergence conditions are met for the sample-average case, $\alpha_k(a) = \frac{1}{k}$, but not for the case of constant step-size parameter, $\alpha_k(a) = \alpha$.

6 Quick Links

- Temporal difference learning and TD-Gammon