Summary - Chapter 2

Ameet Deshpande

December 2, 2017

1 Summaries

Contrast between purely evaluative and purely instructive feedback. Instructive feedback tells what the right action is, regardless of what the action that was taken was. Purely evaluative feedback gives an absolute measure for the action, but does not indicate relative usefulness like if it was the best or worst actions.

1.1 Section 2.1 - An n-armed bandit problem

- The n-armed bandit problem is expected to have a stationary probability distribution.
- In the problem, each action is expected to have an expected value based on the probability distribution, called the *value* of the arm.
- Greedy actions show the exploiting behavior and non-greedy actions show a exploring behavior.
- Supervised learning methods perform very poorly on these kind of problems as they do not balance the explorations and exploitation. This is probably because they return one arm as the correct answer and are more biased towards exploitation.

1.2 Section 2.2 - Action-Value Methods

- $Q_t(a)$ is used to represent the estimate formed for that arm after t time plays. $Q^*(t)$ denotes the actual or true value for that action. One way to estimate Q_t is to take the running average. The *sample-average* method samples the same action a large number of times to arrive at an accurate estimate, supported by the law of large numbers.
- ϵ greedy methods choose actions greedily most of the times (with probability $1 - \epsilon$) and with ϵ probability, the agent explores. An important advantage of this method is that in the limit of infinite number of plays, all the actions will get picked infinite number of time for non-zero ϵ.
- A good strategy could be to lower the value of ϵ over time. ϵ greedy methods are expected to perform even better than normal methods when the variance in rewards is larger. This is because it will take even more exploration to find the best rewards.

- These methods are especially useful when the underlying distribution is not stationary. In such a case, sufficient exploring allows the agent to catch on to the optimal actions even if they can change over time.

1.3 Section 2.3 - Softmax Action Selection

- Gibbs or Boltzmann distribution is used so that the actions which have higher reward returns are picked more often. The temperature parameter τ controls how soft or hard the softmax function is. $e^{\frac{Q(a)}{\tau}}$ is used for each term.

1.4 Section 2.4 - Evaluation versus Instruction

- In supervised learning, it is somewhat like learning by instruction. The right answer is given beforehand. This is in contrast to learning by selection. The instructive feedback is thus typically independent of what action is chosen. It always returns the right answer.
- A supervised learning system cannot be said to learn to control its environment because it follows, rather than influences, the instructive information it receives.

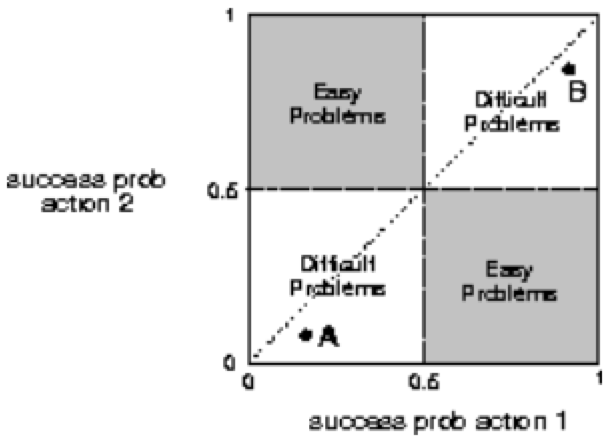

Figure 1: Supervised vs Reinforcement

- In the above figure, supervised learning performs very bad compared to ϵ-greedy methods in the white regions.

1.5 Section 2.5 - Incremental Implementation

- To maintain the average Action-Value estimates, we don't need to store the rewards we received at each time step. Instead we just store the average value at each instant and the number of time steps that have passed.
- Rewriting the rule for average calculation, we get the following formulae.

$$Q_k + \frac{1}{k+1} \times (r_{k+1} - Q_k)$$

This formula can be used to get to general update rules

$$Q_k + \alpha \times (r_{k+1} - Q_k)$$

The formula thus aims to take a step towards the "actual" estimate, which is r_{k+1} by some factor α. Depending on if α is constant or if it decays over time like $\frac{1}{k}$, the final estimate will differ.

1.6 Section 2.6 - Tracking a Non-stationary Problem

- Having a constant α helps in weighting the recent terms/rewards more than rewards which we're received earlier on. It presents a geometric kind of an answer. Not all α guarantee convergence. The conditions for convergence will thus be, $\sum \alpha_k = \inf$ and $\sum \alpha_k^2 < \inf$
- Step-size sequences that meet the conditions often converge very slowly or need considerable tuning in order to obtain a satisfactory convergence rate. This constant α, which does not satisfy these criterion, are considerably used.

1.7 Section 2.7 - Optimistic Initial Values

- It is an easy way to provide or augment prior knowledge.
- Choosing optimistic initial values encourages exploration. Say the expected rewards are 1 and we choose 5 as the initial values. After choosing one of the actions, the reward will definitely fall below 5. This will force the agent to choose other actions whose rewards are still 5. This will allow a lot of exploration even if we are using a completely greedy method.
- The disadvantage of this method is the fact that the drive to explore is temporary. That is, only in the beginning. This is just a simple trick which can be used in the case of stationary problems and will not work in the case of non-stationary problems.

1.8 Section 2.8 - Reinforcement Comparison

- In the methods discussed so far, all the rewards are estimated in the absolute sense. When a reward is received from the environment for taking an action, there is no way to gauge if it is a good or a bad action. We have to perform a *comparison* to get some knowledge of the same.
- Each action has a preference which is updated based on how close or far away from the average rewards, their current reward is. The average itself also needs to be updated. The update rule for the average can be like the following.

$$r\_mean_{t+1} = r\_mean_t + \alpha \times (r_t - r\_mean_t)$$

- In the above update rule, it would actually be a good idea to keep a constant α because as the agent learns, it will learn to choose the correct actions and the rewards will gradually increase. Thus we would want to give higher weight to rewards it has received more recently.

1.9 Section 2.9 - Pursuit Methods

- Look at highlighted portions of text

1.10 Section 2.10 - Associative Search

- As an example, suppose there are several different n -armed bandit tasks, and that on each play you confront one of these chosen at random. Thus, the bandit task changes randomly from play to play. This would appear to you as a single, non-stationary n -armed bandit task, whose true action values change randomly from play to play. You could try using one of the methods described in this chapter that can handle non-stationarity, but unless the true action values change slowly, these methods will not work very well.
- Associative search tasks are intermediate between the n -armed bandit problem and the full reinforcement learning problem. They are like the full reinforcement learning problem in that they involve learning a policy, but like the n -armed bandit problem in that each action affects only the immediate reward

2 Section 2.11 - Conclusion

- One idea to encourage exploration is to use uncertainty in estimating the rewards for the actions. If the reward estimate is pretty certain for an action (maybe it has been chosen multiple times), there is probably no need to take that action again and again. If some action has rewards that is uncertain, we might want to choose that action and explore.
- Interval estimation methods estimate values as intervals rather than values. Instead of saying that the value is 10, which is albeit an estimate, it return the value as being between $9 - 11$ with 95% confidence.
- The classical solution to balancing exploration and exploitation in n -armed bandit problems is to compute special functions called Gittins indices. These provide an optimal solution to a certain kind of bandit problem more general than that considered here, but which assumes the prior distribution of possible problems is known.

3 Exercises

3.1 Exercise 2.1

In the comparison shown in Figure 2.1, which method will perform best in the long run, in terms of cumulative reward and cumulative probability of selecting the best action? How much better will it be?

The method with $\epsilon = 0.01$ will be expected to do better in terms of cumulative probability in the long run as it's average rewards will be higher than that of $\epsilon = 0.1$ as it will be playing optimally most of the time after discovering what the optimal action is. 99.1% vs 91%.

3.2 Exercise 2.2

Programming Check GitHub for Code

3.3 Exercise 2.3

Show that in the case of two actions, the softmax operation using the Gibbs distribution becomes the logistic, or sigmoid, function commonly used in artificial neural networks. What effect does the temperature parameter have on the function?

After writing down the Gibbs terms for both the arms, making the numerator 1 by multiplying the numerator and the denominator by $e^{\frac{-x}{\tau}}$ will give the familiar logistic function.

3.4 Exercise 2.4

Consider a simplified supervised learning problem in which there is only one situation (input pattern) and two actions. One action, say a, is correct and the other, b, is incorrect. The instruction signal is noisy: it instructs the wrong action with probability p; that is, with probability p it says that b is correct. You can think of this as a binary bandit problem if you treat agreeing with the (possibly wrong) instruction signal as success, and disagreeing with it as failure. Discuss the resulting class of binary bandit problems. Is anything special about these problems? How does the supervised algorithm perform on this type of problem?

Do this problem again.

Expected reward after sufficient number of plays will be $1-p$ and p for a and b respectively. The underlying reward function is clearly a Bernoulli probability distribution, with probability $1-p$ for a to get a reward equivalent to *success* (say 1). But more interesting here is that the probability distributions are not independent of each other. Say X and Y are the Bernoulli Random Variables. There is a constraint that $X + Y = 1$. This is thus a special problem because the probability distributions underneath are not independent for different arms.

3.5 Exercise 2.5

Programming

3.6 Exercise 2.6

If the step sizes, , are not constant, then the estimate is a weighted average of previously received rewards with a weighting different from that given by (2.6). What is the weighting on each prior reward for the general case?

Like in the constant α case, just writing down the terms and expanding will give us the correct results. The αs are not constant at each time instant now. The final formulae should come out to be something like

$$\sum r_{k-i} \times \alpha_{k-i} \times \prod (1-\alpha_r)$$

where r will go from $k-i+1$ to k

3.7 Exercise 2.7

Programming

3.8 Exercise 2.8

The results shown in Figure 2.4 should be quite reliable because they are averages over 2000 individual, randomly chosen 10-armed bandit tasks. Why, then, are there oscillations and spikes in the early part of the curve for the optimistic method? What might make this method perform particularly better or worse, on average, on particular early plays?

The initial values for all the arms are much greater than actual values. There will thus be fair bit of exploration in the beginning. Thus, even if the optimal action is chosen initially, it will be given up to choose some action which has the optimistic initial estimates. After a long time however, the true best arm is chosen.

3.9 Exercise 2.9

The softmax action-selection rule given for reinforcement-comparison methods (2.8) lacks the temperature parameter, , used in the earlier softmax equation (2.2). Why do you think this was done? Has any important flexibility been lost here by omitting ?

The introduced parameter β gives us the same amount of flexibility that the parameter τ can give. Thus no flexibility is lost.

3.10 Exercise 2.10

The reinforcement-comparison methods described here have two step-size parameters, and . Could we, in general, reduce this to just one parameter by choosing ? What would be lost by doing this?

α and β are serving similar purposes. Both of them are controlling the rate of convergence for the estimate. Enforcing $\alpha = \beta$ will force the rate of convergence to be the same, which does not make sense. Also, β is constrained by the fact that it is estimating probabilities. Thus, it can take only values such that $p_t(a)$ is between 0 and 1.

3.11 Exercise 2.11

Programming

3.12 Exercise 2.12

An ϵ-greedy method always selects a random action on a fraction, , of the time steps. How about the pursuit algorithm? Will it eventually select the optimal action with probability approaching certainty?

Borrowed Answer

The pursuit algorithm is not guaranteed to always select the optimal action. For example if the initial rewards are such that we incorrectly update the wrong actions, causing $\pi(a)$ to become biased away from its true value we could never actually draw from the optimal arm. Setting the initial action probabilities equal to a uniform distribution helps this. In the case where the prior distributions are set such that $\pi = 1$ for a specific arm and $\pi = 0$ for all others emphasizes the fact that the algorithm does not fully explore the state space.

3.13 Exercise 2.16

Suppose you face a binary bandit task whose true action values change randomly from play to play. Specifically, suppose that for any play the true values of actions 1 and 2 are respectively .1 and .2 with probability .5 (case A), and .9 and .8 with probability .5 (case B). If you are not able to tell which case you face at any play, what is the best expectation of success you can achieve and how should you behave to achieve it? Now suppose that on each play you are told if you are facing case A or case B (although you still don't know the true action values). This is an associative search task. What is the best expectation of success you can achieve in this task, and how should you behave to achieve it?

For both the arms, the expected reward is 0.5. If we are told which case we will be facing, it will be $\frac{0.2+0.9}{2} = 0.55$. This is assuming that we face both the cases A and B with equal probabilities.