Autonomous and Adaptive Systems

Multi-Armed Bandits

Mirco Musolesi

mircomusolesi@acm.org

Credit: Carl Raw

Multi-armed Bandits

- We will start by considering a simplified version of reinforcement learning called multi-armed bandits.
- In the case of a multi-armed bandits, no state is used for the selection of the next actions.
- We will then consider an “intermediate case”, where the state is used but the state itself does not depend on the previous actions, called “contextual bandits”.

Multi-Armed Bandits, Contextual Bandits and Reinforcement Learning

No state is used	Multi-Armed Bandits
State is used State does not depend on previous actions	Contextual Bandits
State is used State depends on previous actions	Reinforcement Learning

Multi-Armed Bandits, Contextual Bandits and Reinforcement Learning: Alternative View

No state is used	Multi-Armed Bandits
State is used **The action that is selected does not affect the next state/situation.**	Contextual Bandits
State is used **The action that is selected affects the next state/ situation.**	Reinforcement Learning

k-armed Bandit Problem

- We can model a k-armed bandit problem as follows:
  - k different actions;
  - After each choice you receive a numerical value that depends only on the action you selected.
- The goal is to maximise the expected reward over a certain number of time steps.
- This is the "classic" k-armed bandit problem, which is named by analogy to a slot machine or "one-armed" bandit, except it has k levers instead than one.
  - Like in a slot machine, we need to learn which lever provides us with the highest reward.

k-armed Bandit Problem

- The problem is the same, maximising:

$$
\begin{aligned}
E[G_t \,|\, S_t = s, A_t = a] &\doteq E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \,|\, S_t = s, A_t = a] \\
&= E[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,|\, S_t = s, A_t = a]
\end{aligned}
$$

- In the case of the k-armed bandit problem, the state is always the same (or, in other words, it does not matter). You can think about having $s = \bar{s}$ with $\bar{s}$ constant.

- In other words the problem is equal to maximise:

$$
\begin{aligned}
E[G_t \,|\, S_t = \bar{s}, A_t = a] &\doteq E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \,|\, S_t = \bar{s}, A_t = a] \\
&= E[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,|\, S_t = \bar{s}, A_t = a]
\end{aligned}
$$

with a one of the k actions.

k-armed Bandit Problem

- In a k-armed bandit problem, each of the k actions has a value, equal to the expected or mean reward given that that action is selected.
- As in the general RL case, we denote the action selected on time step t as A_t and the corresponding reward as R_t.
- The value of an arbitrary action a denoted as $q_*(a)$ is the expected reward given that a is selected. More formally:

$$q_*(a) \doteq \mathbf{E}[R_t | A_t = a]$$

k-armed Bandit Problem

- Trivial case: we know the value of each action, then solving the k-armed bandit problem is very easy:
  - It is sufficient to always select the action with the highest value!
- However, in general, we do *not* know the action values with certainty, we only know *estimates*.
- We denote the estimated value of action a at time step t as $Q_t(a)$.
- Ideally, we would like to have that the value of $Q_t(a)$ would be very close to $q_*(a)$.

Exploration vs Exploitation

- We maintain the estimates of each action value.
- At any step there is at least one action whose estimated value is the greatest (we refer to this action as *greedy action*).
- When we select one of these actions, we are exploiting our current knowledge of the values of the actions (exploitation).
- If we select non-greedy actions, we say that we are *exploring* (alternative actions). By doing so, we can improve our estimation of the value functions of the non-greedy actions.
- Balancing exploration and exploitation in a smart way is the key aspect.
  - Several theoretical results in terms of bounds, etc. given specific assumptions.

Evaluating Action-value Methods

- Multi-armed bandits are a very good way of approaching the general problem of Reinforcement Learning.
- Simplification: the state does not change, which means:
    - Your actions do not modify the state;
    - Since the state does not change, the agent's actions will not depend on the previous actions (the two things are strictly linked).
- We will consider later the "full" reinforcement learning problem where the agent's actions *do* modify the state and the agent's action *do* depend on the previous actions.

Evaluating Action-value Methods

- Recall: the true value of an action is the mean reward when that action is selected.
- A possible way to estimate this is by averaging the rewards actually received:

$$Q_t(a) \doteq \frac{sum\ of\ rewards\ when\ an\ action\ a\ is\ taken\ prior\ time\ t}{number\ of\ times\ an\ action\ a\ is\ taken\ prior\ time\ t}$$

$$= \frac{\sum_{i=1}^{t-1} R_i 1_{A_i=a}}{\sum_{i=1}^{t-1} 1_{A_i=a}}$$

where 1 denotes the random variables that is 1 if the predicate $A_i = a$ is true and 0 if not. If the denominator is 0, we set $Q_t(a)$ to some default value (e.g., 0).

Evaluating Action-value Methods

- As the denominator goes to infinity, by the law of large numbers, $Q_t(a)$ converges to $q_*(a)$.
- This is called the *sample-average method*, because each estimate is the average of the sample of the rewards.

Greedy Action Selection Rule

- The simplest selection rule is to select one of the actions with the highest estimated value. This is usually referred to as greedy selection rule.

- More formally, a greedy selection rule is one that selects A_t so that:

$$A_t \doteq \arg\max_a Q_t(a)$$

i.e., the action a for which $Q_t(a)$ is maximised.

ϵ-greedy Selection Rule

- A simple alternative is to behave greedily most of time, but from time to time, with a probability ϵ select instead randomly from among all the actions with equal probability. We call this method ϵ-greedy.

- One of the advantages of the method is that, in the limit, as the number of steps increases, all the actions will be sampled an infinite amount of time, that ensuring that for all a the $Q_t(a)$ will converge to $q_*(a)$.

Incremental Implementation

- We now discuss how to implement these methods in practice.
- To simplify the notation, we consider on a single action a.
- Let $R_i(a)$ the reward received after the ith selection of the action a.
- Let $Q_n(a)$ denote the estimate of its action value after it has been selected $n-1$ times.
- We can write:

$$Q_n(a) \doteq \frac{R_1(a) + R_2(a) + \ldots + R_{n-1}(a)}{n-1}$$

Incremental Implementation

- The trivial implementation would be to maintain a record of all the rewards and the execute the formula when that value would be needed.
- However, if this is done, the computational requirements would grow over time.
- Possible alternative: an incremental implementation.

Incremental Implementation

$$
\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}(R_n + \sum_{i=1}^{n-1} R_i) \\
&= \frac{1}{n}(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i) \\
&= \frac{1}{n}(R_n + (n-1)Q_n) \\
&= \frac{1}{n}(R_n + nQ_n - Q_n) \\
&= Q_n + \frac{1}{n}(R_n - Q_n)
\end{aligned}
$$

Incremental Implementation

- This form of update rule is quite common in reinforcement learning.
- The general form is as follows:

NewEstimate <- OldEstimate + StepSize (Target - OldEstimate)

- The expression *(Target-OldEstimate)* is usually defined as error in the estimate.

Tracking a Nonstationary Problem

- The averaging method discussed before is appropriate for stationary bandit problems.
- However, many problems are *non-stationary*.
- In these cases, it makes sense to give more weight to recent rewards rather than long-past rewards.
- One of the most popular way of doing this is to use constant step-size parameter (in the example above was $\frac{1}{n}$).

Tracking a Nonstationary Problem

- The incremental update rule for updating an average Q_n of the $n-1$past rewards is modified to be:

$$Q_{n+1} \doteq Q_n + \alpha(R_n - Q_n)$$

where the step-size parameter $\alpha \in (0,1]$ is constant.

- This results in Q_{n+1} being a weighted average of past rewards and the initial estimate.

- More formally:

$$Q_{n+1} \doteq Q_n + \alpha(R_n - Q_n) = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{(n-i)} R_i$$

Tracking a Nonstationary Problem

- This is called a weighted average since

$$(1-\alpha)^n + \sum_{i=1}^{n} \alpha(1-\alpha)^{(n-i)} = 1$$

- The quantity $1-\alpha$ is less than 1 and the weight given to R_i decreases as the number of rewards increases (actually it’s exponential, and for this reason it is sometimes called *exponential-recency-weighted average*).

Optimistic Initial Values

- These methods are dependent on the initial action-value estimates $Q_1(a)$.
    - We say that these methods are biased by their initial estimates.
- Initial values can be used as a simple way to encourage exploration.
    - If the learner is “disappointed” by the initial values and explore more.
- For example, if the initial values for each action a are selected in order to be quite high with respect to the actual optimal values, all the actions are tried several times before we get convergence.

Optimistic Initial Values: Example

- Suppose that we have a multi-armed bandit, where the optimal action values $q_*(a)$ for every action a are selected from a normal distribution with mean 0 and variance 1.

- In this case, an initial value $Q_1(a) = 10$ for each activity is highly disappointing.

- Remember the formula:

NewEstimate <- OldEstimate + StepSize (Target - OldEstimate)

- This will lead to a lot of exploration, also in pure greedy action selection.

Upper-Confidence-Bound (UCB) Action Selection

- Exploration is needed because there is always uncertainty of the actual-value estimates.
  - Greedy actions might be the best ones, but it might not be the case.
  - ϵ-greedy action selection forces the non-greedy actions to be tried, but indiscriminately with no preferences for those that are nearly greedy or particularly uncertain.
- It would be better to select among the non-greedy actions considering their potential of being optimal, taking into account both how close their estimates are to being optimal and the uncertainties in these estimates.

Upper-Confidence-Bound (UCB) Action Selection

- One effective way of doing this is to select actions according to:

$$A_t \doteq \arg\max_a \left(Q_t(a) + c\sqrt{\frac{lnt}{N_t(a)}}\right)$$

where t is the time at which action A_t is taken, $N_t(a)$ denotes the number of times that action a has been selected prior to time t and the number $c > 0$ controls the degree of exploration.

- The idea of this upper confidence bound (UCB) action selection is that the square-root term is a measure of the uncertainty or variance of the estimate of the actual value of action a.

Contextual Bandits

- Multi-armed bandits are used when the selection of the action is not dependent on the state.
- But for many applications (e.g., ads on a webpage), actually the state is very important.
- When the action selection depends on the state, but not on its previous history, we have the case of *contextual bandits*.
- Since the action is associated to the state, contextual bandits are also defined as a problem of *associative search*.

Contextual Bandits

- They require to learn a policy, i.e., a mapping from the current state to the probabilities of each action.
- As we said, contextual bandits differ from the “full” reinforcement learning problem in that the action does not affect the future state.
- In case, the action affects the future state, we have the “full” reinforcement learning problem.

Contextual Bandits

- Recall the definition of policy:
  - A *policy* is a mapping from states to probabilities of each possible action.
  - If the agent is following policy π at time t, then $\pi(a \mid s)$ is the probability that $A_t = a$ if $S_t = s$.
- In contextual bandits, the action a that is selected does not modify the state. In other words, the action a does not modify the next situation.
- Methods are similar to those for full reinforcement learning.

Exercise

- Design a system serving ads on a website (e.g., Google Ads) using contextual bandits.

Additional Readings

- Lihong Li, Wei Chu, John Langford, Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. *Proceedings of the 19th International Conference on World Wide Web (WWW 2010)*: 661–670.

References

- Chapter 2 of Sutton and Barto. Introduction to Reinforcement Learning. Second Edition. MIT Press. 2018.
- Tor Lattimore and Czaba Szepesvári. Bandit Algorithms. Cambridge University Press. 2020.