Exploration vs. Exploitation

Krishna Devkota

Bielefeld University
5th May 2017

Reinforcement Learning in Autonomous Social agents

Outline

The Exploration vs. Exploitation dilemma

■ Important aspect of model-free algorithms is a need for exploration

■ As model unknown, learner needs to try out different actions to see their results

■ How can a RL agent balance Exploration vs. Exploitation?

One of the fundamental questions in RL

- Exploration

  Gather more information

- Exploitation

  Make the best decision given current information

■ Sometimes, immediate sacrifices might lead to better long-term strategies

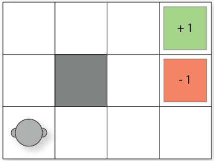
Exploration vs. Exploitation (In Practice)

- Restaurant Selection
  Exploitation: Go to a Restaurant that you know well
  Exploration: Try a new restaurant
- Playing a game
  Exploitation: Play a move that you are confident of
  Exploration: Try a new move that you haven't played much
- Advertisement
  Exploitation: Play an advertisement that has been received well
  Exploration: Try a new Ad that has not been played yet

Quick Recap of Markov Decision Process (MDP)

Markov decision processes formally describe an environment for reinforcement learning (decision making)

- A MDP can be represented as a 4-tuple $\langle S, A, P, R \rangle$ where:



- S is a set of states
- A is set of all the actions that the agent can take
- $P(s'|s, a)$ is a function that defines the transition probability *(Markovian)*
- $R(s, a)$ is the reward function, which gives the probability of receiving reward r after choosing a in state s

Three different classes of approach to the problem

- Random exploration
  - Explore random actions (e.g. $\epsilon - greedy$, softmax)
- Optimism in the face of uncertainty
  - estimate uncertainty on value
  - Prefer to explore states/actions with highest uncertainty
- Information state space
  - Consider agent's information as part of its state
  - Look ahead to see how information helps reward

Random exploration (naive approach)

Multi-armed Bandit Problem (single-state MDP)

One of the simplest way to model a exploration/ exploitation dilemma is using a multi-armed Bandit Problem

Figure: Bandit Machines

Multi-armed bandit problem

- models the exploration/exploitation trade-off inherent in sequential decision problems
- a sequential experiment with the goal of achieving the largest possible reward from a payoff distribution
- Parameters of payoff distribution unknown
- Choice involves a fundamental trade-off between:
  - the utility gain from exploiting arms that appear to be doing well (based on limited sample information)
  - *vs.* exploring arms that might potentially be optimal, but which appear to be inferior because of sampling variability
- sometimes referred to as 'earn vs learn'

Formalizing Multi-armed Bandit problem

- Can be represented as a Tuple $\langle A, R \rangle$
- A is known set of Arms that can be Pulled i.e. actions that can be taken (m)
- $R^a(r) = \mathcal{P}[r|a]$ is the unknown Probability distribution over rewards
- Action taken at each step t by the agent : $a_t \in A$
- Reward generated by the environment: $r_t \sim R^{a_t}$

■ Goal : maximize cumulative reward

$$\sum_{\tau=1}^{t} r_\tau$$

Regret

- Action-value
  mean reward for action a, $Q(a) = E[r|a]$
- Optimal value (V^*)
  $V^* = Q(a^*) = max_{a \in A} Q(a)$
- Regret
  Opportunity lost for each step $l_t = E[V^* - Q(a_t)]$
- Total Regret
  Opportunity lost over all the steps
  $L_t = E\left[\sum_{\tau=1}^{t}(V^* - Q(a_\tau))\right]$

■ Goal : maximize cumulative reward, hence minimize total regret

Regret as counts and gaps

- Count ($N_t(a)$)
  Expected number of times an action is taken,
- Gap ($\triangle_a$)
  Difference in value between action (a) and optimal action (a^*) $\triangle_a = V^* - Q(a)$
- Regret as a function of count and gap
  $L_t = \sum_{a \in A} E\left[N_t(a)\right] \triangle_a$

■ Goal : Find a good algorithm, so we visit the less desired state, the least amount of times i.e. small counts for large gaps

■ Problem : we don't know the optimal value (V^*), and hence the gaps

Regret

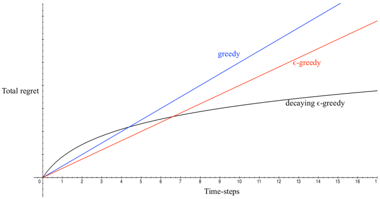


Figure: Comparing Regret values for different naive-algorithms

- If an algorithm forever explores, it will have a linear regret ($\epsilon - greedy$)
- If an algorithm never explores, it will have a linear regret (greedy)
- So, how do we achieve sub-linear total regret?

A quick look at Greedy algorithm

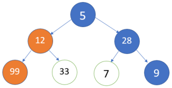


Figure: Greedy algorithm based on local optimum

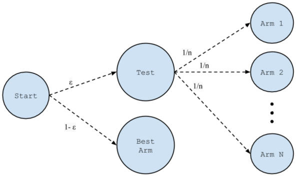
- Consider algorithm that estimates $\hat{Q}_t(a)$ which is closest to $Q_t(a)$ i.e. the MC evaluation:

$$\hat{Q_t}(a) = \frac{1}{N_t(a)} \sum_{t=1}^{T} r_t 1(a_t = a)$$

- Using greedy algorithm gives us: $a_t{}^* = argmax_{a \in A} \hat{Q}_t(a)$
- Problem: We might get stuck onto a suboptimal action again and again

Outline

The $\epsilon - greedy$ algorithm



- Using the $\epsilon - greedy$ algorithm, we want to introduce some randomness into our greedy approach
- How do we do that??
  - with probability $1 - \epsilon$ act greedily, i.e. select $a_t{}^* = argmax_{a \in A} \hat{Q}_t(a)$
  - with probability ϵ, select a random action

Advantages of $\epsilon - greedy$ exploration:

- Simplest idea for ensuring continual exploration
- All m-actions are tried with non-zero probability

Drawback of $\epsilon - greedy$ exploration:

- Random actions selected uniformly. The worst possible action is just as likely to be selected as the second best action

Outline

Softmax

We saw that the $\epsilon - greedy$ selected the random actions uniformly, giving equal weights to the good and the bad Softmax remedies this by assigning a rank or weight to each of the actions, according to their action-value estimate.

- Bias exploration towards promising actions
- Softmax action selection methods grade action probabilities by estimated values
- The most common softmax uses a Gibbs (or Boltzmann) distribution

Advantages:

- As appropriate weight associated with each action, the worst actions are unlikely to be chosen
- Good in scenarios where the worst actions are very unfavourable

Softmax

******************** Mathematical formulation *********************

Outline

Optimistic Initialization

- initialize Q(a) to high value (i.e. assume all of our actions pay the best possible

$$Q(a) = r_{max}$$

- Use MC evaluation to incrementally update action value
- $\hat{Q}_t(a_t) = \hat{Q}_{t-1} + \frac{1}{N_t(a_t)}(r_t - \hat{Q}_{t-1})$

Advantage:

- Encourages exploration of unknown values

Drawback:

- We need to know the maximum possible reward r_{max}
- Can still get caught in suboptimal action

Comparing greedy, ϵ-greedy, and Optimistic Initialization

For a 10-armed testbed, N = 10 possible actions, 1000 plays
Q(a) are chosen randomly from a Normal distribution $N(0, 1)$

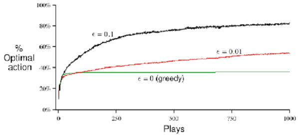


Figure: Greedy vs. ϵ-greedy

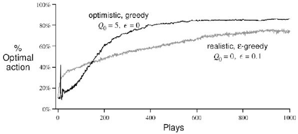
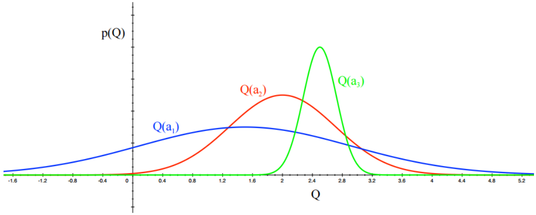


Figure: Normal case vs. Optimistically initialized case

Methods based on Upper Confidence Bounds (UCBs)

Acting Optimistically in Uncertain situations



- The more uncertain we are about an action value
- The more important it is to explore that action
- It could turn out to be the best action

Outline

Methods based on Upper Confidence Bounds (UCBs)

- So, far we've talked about estimating the mean (when we talk about Q-values)
- i.e. which arm gives the max reward on average
- but we don't know the uncertainty in each arm
- the range of values that the arm can give

■ We can assume two different approaches to address this

- Frequentist:
  where we assume nothing about the distribution
- Bayesian:
  assume we have some prior probabilities over the Q-values

Methods based on Upper Confidence Bounds (UCBs)

- Confidence interval
  a range of values within which we are sure the mean lies with a certain probability
- For an action which has been tried less often, our estimated reward is less accurate so the confidence interval is larger
- It shrinks as we get more information (i.e. try the action more often)
- So, instead of trying the action with the highest mean, we can try the action with the highest upper bound on its confidence interval
- This is known as an optimistic policy

Methods based on Upper Confidence Bounds (UCBs)

Steps:

- For each action value, estimate an upper confidence $\hat{U}_t(a)$ such that:
  $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with high probability
- determined by the number of times N(a) has been selected
  - Small $N_t(a) \Rightarrow large \hat{U}_t(a)$
    estimated value is uncertain
  - Large $N_t(a) \Rightarrow small \hat{U}_t(a)$
    estimated value is accurate
- Select action that maximizes the UCB
  $a_t = argmax_{a \in A} \hat{Q}_t(a) + \hat{U}_t(a)$

Methods based on Upper Confidence Bounds (UCBs)

- To solve for the bounds, we can turn to: Chernoff-Hoeffding bound
- Then, pick a probability p for the true value to exceed the UCB
- Solve for $U_t(a)$, and reducing probability p, as more rewards are observed
- As $t \to \infty$, we select optimal action as given by:

$$U_t(a) = \sqrt{\frac{2logt}{N_t(a)}}$$

This gives us the UCB1 algorithm:

$$a_t = argmax_{a \in A} Q(a) + \sqrt{\frac{2logt}{N_t(a)}}$$

Quick recap of methods based on UCB

- Each arm is assigned an UCB for its mean reward
- Arm with the largest bound to be played
- Bound is not conventional upper limit for a confidence interval, hence difficult to compute
- However, making some basic assumptions, the expected number of times suboptimal arm a would be played by time t is:
  $E(n_{at}) \leq \left(\frac{1}{K(a,a^*)} + o(1)\right) logt$
  where $K(a, a^*)$ is the Kullback-Leibler divergence between the reward distributions for arm a and optimal arm a^*
- This bound essentially says that the optimal arm will be played exponentially more often than any of the suboptimal arms, for large t

How do we exploit prior knowledge about rewards?

- Recall, we started by a unknown distribution over our action-value function
- Instead, say we start with some distribution over the action value function
- Let $p[Q|w]$ be some distribution over action-value function, where w is the parameter
- The parameters w could be (say) the mean and the variances of each of our arms
- We could then compute posterior distribution over w by using the Bayesian methods $p[w|R_1, ..., R_t]$

- the posterior can then be used to guide exploration i.e. UCB, and Probability matching
- the performance is better if our knowledge of the prior is accurate

Random Probability Matching

- Randomized probability matching combines many positive aspects of the heuristic strategies mentioned above
- Probability matching selects action a according to probability that a is the optimal action $\pi(a|h_t) = P[Q(a) > Q(a'), \forall a' \neq a|h_t]$
- Uncertain actions have higher probability of being max
- Can be difficult to compute analytically from posterior

Outline

Thompson sampling

- way to implement probability matching
  $\pi(a|h_t) = P[Q(a) > Q(a'), \forall a' \neq a|h_t]$
  $= E_{R|h_t}\left[1(a = argmax_{a \in A} Q(a))\right]$
- Use Bayes law to compute posterior distribution $p[R|h_t]$, where $h_t = a_1, r_1, ..., a_{t-1}, r_{t-1}$ is the history
- Sample a reward distribution R from posterior
- Compute action-value function $Q(a) = E[R_a]$
- Select action maximising value on sample, $a_t = argmax_{a \in A} Q(a)$

Advantages of Probability matching techniques:

- the tuning parameters, and the decay schedule evolves in a principled, data-determined way
- In other methods, the parameters are arbitrarily set by analyst, and incorrect values bear huge costs

Disavantages of Probability matching techniques:

- There is a need to sample from the posterior distribution
- This can require substantially more computing than other heuristics

Outline

Outline

Summary

■ We saw how the problem of exploration/ exploitation can be tricky sometimes

■ We looked at some of the heuristic strategies to handle the dilemma (naive)

- Equal allocation
- Play-the-winner
- Deterministic greedy strategies
- Hybrid strategies such as $\epsilon - greedy$, and Softmax

■ We looked at some strategies based on upper bounds

- UCB1
- Random Probability matching

*************Summary goes here**************

For Further Reading I

A. Author.
Handbook of Everything.
Some Press, 1990.

S. Someone.
On this and that.
Journal of This and That, 2(1):50–100, 2000.