Investigation of the Restless Multi-Armed Bandit Problem

Supervised Research Exposition

Guide: Prof. D.Manjunath

Goutham Ramakrishnan, 140020039

Autumn 2016

The traditional multi-armed bandit problem has been extensively studied and has practical applications in problems such as resource allocation. A variation of the multi-armed bandit problem was investigated in this project, which has potential applications in scheduling problems.

Problem Intuition

There are a number of arms(say K), each of which give us a reward upon pulling. We need to find the optimal way to fill T time slots, in each of which we can choose to pull exactly one of the arms, or none at all, such that the total reward obtained in maximized.

Define the state of an arm as the number of time slots since the arm was last pulled. Denote the state of the i^{th} arm by τ_i. Assume that the initial state of all arms is 0. The reward obtained upon pulling upon an arm varies as a function of its state. Hence, pulling an arm resets its state back to 0, while it increases the state of all other arms by 1. It is assumed that the reward functions corresponding to each of the arms is previously known.

Thus, the problem differs from the traditional multi-armed bandit in the following ways: the reward functions are previously known, the state transitions are deterministic and restless , i.e. the states of the arms change even when the arm is not played, but we can find exactly the final state of each arm.

Formal Problem Statement

Given K discrete functions $f_1(\tau_1), \ldots, f_k(\tau_k)$; $\tau_i \in \mathbb{W}$, we need to find a sequence $A = \{a_1, a_2, \ldots, a_T\}$, where $a_i \in \{0, 1, 2, \ldots, K\}$, which maximizes the function:

$$R = f_{a_1}(\tau_{a_1,1}) + f_{a_2}(\tau_{a_2,2}) + \ldots + f_{a_T}(\tau_{a_T,T}) = \sum_{a_i \in A} f_{a_i}(\tau_{a_i,i})$$

where $\tau_{a_i,i}$ denotes the value of τ_{a_i} just before the i^{th} time instant.
$f_0(\tau_0)$ is trivially defined to be zero for all τ_0.
Consider the state of the i^{th} arm just before the t^{th} time instant($\tau_{i,t}$). Let t_i be the largest number such that $t_i<t$ and $a_{t_i} = i$, and zero if no such t_i exists.
Then, the parameter τ_i corresponding to each f_i just before the t^{th} time instant is given by:

$$\tau_{i,t} = t - t_i - 1$$

Reduction to the Knapsack Problem

Sir had a strong intuition that the problem can be reduced to the well-known knapsack problem. As it turned out, he discovered that in the case of a single arm, that is indeed the case. The problem then reduces to packing the knapsack/sequence with sub-sequences of known cost/τ and value/reward. We wish to find the packing scheme which would the maximum reward, subject to the maximum cost constraint represented by the length of the sequence T. However, no such reductions could be thought of for the multi-arm case.

NP-Hardness

The knapsack problem is known to be NP-Complete. Our problem in its simplest form reduces to the knapsack problem. Also, there is no polynomial time algorithm which we could come up with through which a given "witness" sequence can be verified to be the optimum sequence. Hence, it is reasonable to suspect that the problem is NP-Hard.

Reduction from existing NP-Hard problems proved to be difficult, largely due to the large number of parameters in our problem. In any case, this was not the major focus of the project, which was to find a good algorithm to solve the problem.

Optimal Solution

The optimal sequence of arms can be found through dynamic programming. Let Γ be the vector containing the states of all the arms. Define $R(t, \Gamma)$ to be the maximum reward that can be obtained by filling the slots (t,...,T), if the state after the $(t-1)^{th}$ slot is Γ. Then the Dynamic Programming Equation is:

$$R(t, \Gamma) = \max_{i=0,...,K} R(t+1, \Gamma_i)$$

Γ_i is the new state upon playing arm i when the initial state is Γ. $\Gamma_i = (\Gamma + I) \odot I_i$ where $\odot$ is the Hadamard product, I is a K-dimensional vector with all ones, and I_i is I with the i^{th} element as 0.

The optimum reward can therefore be found by building the sequence from the back. The maximum reward must be calculated and stored for every possible Γ, at each time step. We can obtain the maximum reward by obtaining the value of $R(1,\Gamma_{initial})$, where $\Gamma_{initial}$ is the initial state vector(i.e. vector of all zeros).

Theoretically, there are a total of t^K possible state vectors before the t^{th} time instant. However a large number of these states are not attainable due to the nature of the problem. The vectors in the attainable state space are constrained by the following rule: For valid initial states before the t^{th} time instant, no two elements in the state vector can be equal unless they are both equal to $t-1$. For example, before the 5^{th} time slot, (3,3,4) is not a valid initial state whereas (4,4,3) is(attained by playing the sequence {3,0,0,0}).

Thus, the total number of states for computation needs to be performed for the a sequence of T time slots reduces from $\sum_{t=1}^{T} t^K$ to:

$$\sum_{t=1}^{T} \sum_{n=0}^{K} \frac{^{t-1}C_{k-n} k!}{n!}$$

^nC_m is defined to be zero for $m > n$. The above quantity still grows exponentially with K, but can lead to significant computational savings.

Due to the exponentially large state space, the algorithm had a complexity $\mathcal{O}(T^{K+1})$. In the small-K, large-T domain, the algorithm performs well. But, it is exponential in the number of arms. Ideally, we would want the algorithm to be polynomial in both T and K. Despite its shortcomings, this approach provided a way to evaluate and test the performance of other algorithmic approaches.

Assumptions on reward functions

The open nature of the problem allowed making appropriate but justified assumptions regarding the reward functions, for proving the efficacy of an algorithm. Some of these assumptions included bitonic, bounded derivative and piece-wise linear. Bitonic functions seemed to be a reasonable assumption in the context of the problem. Combining this with the piece-wise linear assumption turned out to be an oversimplification.

Alternative Algorithmic Approaches

Several algorithmic approaches were tried in search of an optimal/approximation algorithm of polynomial complexity in both the number of arms and the number of time slots. Define the approximation ratio as the ratio of the reward obtained by an algorithm and the optimal reward. All algorithms were implemented and tested using Python.

- Greedy:

  Greedy algorithms were one of the first approaches to be tried and discarded. No greedy scheme of playing arms which we could come up with could be proven to have a worst-case approximation ratio, let alone optimal. Besides, the very nature of the problem suggests that a greedy solution may not be the correct approach.

- Divide-and-Conquer:

  It is obvious that the reward obtained upon pulling an arm and its corresponding parameter τ represent a trade-off. The reward obtained is the benefit obtained upon paying the cost of refraining from playing the arm for a number of time slots. This trade-off can be represented by a simple ratio of the two.

  An algorithm was devised which recursively built sub-sequences on the basis of the above ratio. The algorithm performed satisfactorily in some test cases, but in some cases its approximation ratio was extremely low. The efficacy of the algorithm could not be analyzed effectively.

- Randomized:

  A randomized approach was employed to arrive at a sequence of arms to be played. For any given state, the probability that an arm would be pulled was proportional to be reward that would be obtained. This resolved some of the issues faced by the greedy approach. However, it posed some poignant issues, such as what should be the probability of no arm being played in a slot?

  Its performance was measured in terms of the average approximation ratio obtained. Depending upon the structure of the reward functions, the performance of the algorithm varied from poor to excellent. It did however show asymptotic convergence to a particular approximation ratio in some cases, as the length of the sequence increased. Once again, the efficacy of the algorithm could not be proved.

- Merging individual optimum sequences:

  The optimum sequences can be obtained for each arm to individually fill the sequence. Attempts were made to merge the obtained individual sequences to obtain the sequence in which the arms should be pulled. This was approached as a shifting and fitting problem. Restrictions were placed on maximum number of times an arm could be played. This approach had obvious flaws, and did not perform satisfactorily.

- Periodic playing of arms:

  The structure of the problem suggests that an optimal policy could involve pulling arms periodically. Intuitively, the optimum state at which an arm should be pulled is the state at which the ratio of reward to τ is maximum. However, analysis of the optimum sequences revealed no such pattern.

  The periodic pulling of an arm does not hold even in the case of a single arm. However, if the length of the sequence is infinite, then it can be proved that an optimum period exists and is equal to the one described above.

$$\text{Optimum Period} = \underset{\tau}{\operatorname{argmax}} \frac{Reward(\tau)}{\tau+1}$$

- Gittins Index:

  The Gittins Index theorem gives an optimal policy to solve the traditional multi-armed bandit problem. This was the inspiration to try and adapt the approach to our problem. Several questions can be raised regarding the validity of this approach to our problem, but it was investigated nevertheless.

  The candidate "index" considered for analysis was obtained by the following procedure. Not playing an arm was given a reward. The optimum periods were found with this incentive in place. The optimum period would then be given by:

  $$\text{Optimum Period} = \underset{\tau}{\text{argmax}} \ \frac{Reward(\tau)+\tau I}{\tau+1}$$

  where I is the incentive to not play any arm.

  The index of the i^{th} arm in a particular state τ_i was defined to be the smallest incentive I that needs to be offered for not playing the arm, such that the optimum period exceeds τ_i. At each time instant, the arm with the largest index is played.

  The algorithm performed well in some cases and poorly in others. Proving the efficacy of the algorithm looks daunting. It may well be worth investigating this approach further.

Conclusion

A number of different approaches to the problem were investigated over a period of many months, and no satisfactory solution has emerged. However, our understanding has certainly been enhanced, and the work done in this project will surely help any future research of the problem.

Though the results obtained weren't as hoped, I thoroughly enjoyed the working on the project. It gave me a glimpse into the difficulties faced in research, and the combination of theory and experimentation it offered suited me well. Doing this project has expanded my knowledge in algorithms, and has encouraged me to think outside the box. The weekly discussions with Prof. Manjunath were lively and interesting. I am grateful to him for the opportunity to work on the project, and for being such a wonderful mentor.