Details of method and some discussion

Considering linear corelation for fast bandit explore

Jia-Qi Yang[1] and Shao-Tong Luan[2]

[1]Nanjing University, Nanjing, China
[2]National University of Defense Technology, Changsha, China
thyrixyang@gmail.com

Thursday 4th July, 2019

1 Method

1.1 Methods in Tutorial

1.1.1 Q learning, DDPG and Policy gradient

This is a continuous environment, so the action must be discretized. The state, however, is very large and can't be approximate very well (as discussed in section 2), so plain Q learning (DDPG as well) is bound to overfit on a very small dataset.

Policy gradient is an on-policy RL algorithm, so it's sample efficiency is much worse than Q learning or DDPG, thus not working as Q learning failed.

1.1.2 Genetic Algorithm and Bandit Randomized Probability Matching

GA is one of the most promising algorithm we think at the beginning of the contest, but it's not working well. The reason is that actions are not local, as in TSP like problem, that swap some values will only affect nearby years. In this problem, action of the first year may largely affect reward of last years. Another problem is that GA needs large population, which is infeasible.

BRPM is somewhat similar with PG, since it's also learning a model of action. BRPM introduced two critical hypothesis: 1. actions of different years are independent 2. actions are single modal (the beta function has only one peak, even if there maybe a equally good but different action). The two hypothesis reduced possible action space, whether it work depends on the correctness of it's hypothesis.

BRPM is the best algorithm among tutorials in our experiment, other algorithms maybe tuned to be better, but they are also more difficult to tune than BRPM.

1.2 Proposed method

We have tried and tuned all methods provided in tutorials, we also tried cross entropy method (CEM), DDPG, kernel density estimation (KDE) and some of their variants, finally come up with the following method that will overcome their disadvantages (hopefully).

Our algorithm is indeed a mixture of Q learning and simulated annealing (SA).

We perform learning on continuous observation and actions, don't use discretization.

1.2.1 Two key points in our method

1. Linear corelation between some actions exists: In this problem, the most significant corelation is ITN and IRS with the same year, if ITN is higher then IRS can be lower, and the

sum of ITN and IRS can't be too high to save money. Other corelations are also possible, for example, if some ITN remain effective in the next year, then there will be a strong corelation between adjacent years.

2. Change related actions in a consistent direction may be better: As we don't know these corelations in advance (which can be hand coded, but we consider this approach as cheating, because if the action space of bandit problem is permuted, won't work at all; but our method won't be affected), we can't easily utilize this knowledge. And there may be some unknown relationships. Our approach is to find out most related actions, and optimize over these actions.

1.2.2 How to measure corelation?

We denote actions as $[x_1, x_2, x_3, ..x_{10}]$.

While linear corelation can be calculated directly, we use a different method.

First, we sample a k subset of actions, $[x_{a_1}, x_{a_2}, ..., x_{a_k}]$, $a_j \in [1, 10]$. Then for these k actions, we have n observations, form a matrix $X^{n \times k}, y$, y is the observed scores. (similar to bagging that we dropped some features).

Then we use a linear regress to model relations between these observations and scores. If we can get a good prediction using these actions, then these actions(features) are most related to score and to each other.

1.2.3 How to explore a (potentially) better action?

The linear relation also gives us an approach to adjust policy.

For example, if we know action 0, 1, 3 is most related, then the linear model can be a good depict of the reward trend (like Q function in Q learning). We can utilize this model to find a good action in mind (but maybe bad, then the model will adjust by itself to correct this error), then query the env to return reward of this policy. We only perform optimization on these actions, which makes it similar to SA or coordinate descent.

In practice, we use an ensemble of a few linear models built on different feature sets to increase stability. We don't optimize precisely, which will lead to over optimistic, instead we sample some actions, then choose the most promising. Our approach incorporates exploitation and exploration naturally.

1.2.4 Advantages of our method

1. Find and optimize over corelated actions automatically.

2. Using feature-wise bagging to deal with overfitting.

3. No assumption on single modal like in BRPM; Don't suffer from the instability and infeasibility of cross operation in GA.

4. Can find relatively good solution more frequently compared to other approach, very bad outcome (e.g. negative reward) are less likely.

5. While our algorithm are more stable than random explore, this may harm the ability to find very high reward, so the average reward may be a better indicator of performance rather than medium for our algorithm.

1.2.5 Hyper-parameters

Warm up trial number: how many random trials we need to gather enough relation information.

Model number: how many feature set need to be tried.

Feature number: select how many feature out of 10 features.

Survive number: Use how many feature in the ensemble model.

Sample number: how many actions are tried on trained model.

These are all parameters, we didn't spend much time on tuning these parameters, but we expect our method is robust to these hyper-parameters.

Tune these parameters may lead to better result.

1.3 Extend to a theoretical paper

As far as I know, there are few theoretical results on explore efficiency in such constrained trial bandit problem.

This method, however, may not work very well on the final test environment, because of the randomness of the score mechanism, and the "no free lunch theorem". The environment will change a lot in the final test, it can hardly be treated as the same problem (this change maybe larger than from an atari game to another), which makes a problem specific design harder, so We didn't spend much time on tuning hyper-parameters.

Under some assumptions that the environment is locally linear and convex, with some random noise, our method that utilizing linear dependency may give a tighter bound on convergence, more experiments are also needed, which will be a future work.

2 About the problem setting

We are in full agreement on the problem setting of this competition, especially on the constraint of number of trials, which makes this problem totally different from any RL problem We've met before.

20 trials for Malaria Control, however, still seems luxurious, because this experiment will take on thousands of people for 100 years.

Exploring an action space of size $[0, 1]^{10}$ with in 20 trials is theoretically impossible, because even if we only search for discrete value 0 and 1, there are still $2^{10} = 1024$ possible policies.

The environment is not a MDP if we take actions of previous year as state, so the observation space is large even when we go step by step. Because if we only consider 5 values 0, 0.25, 0.5, 0.75, 1, the observation space is 25 in the second year, and 125 in the third, which make tabular Q learning infeasible.

On the other hand, the reward function of just the first year is already very complex and rugged, so can't be approximated very well in only 20 trials. The error will accumulate later and make precision much worse. Taking this into account, we didn't use sequential information.

We think there are indeed some probability that the sequential information can help in this problem, if there is a way to utilize some problem specific structure (which we didn't find out, for example, the 4th year may highly depend on the action of the 2nd year) in this problem.

However, any algorithm that seems perform better constantly (if it's not an outcome of randomness) in this problem should be treated carefully: It's impossible to be better than random without any assumption (because we can't try every possible policies), and if there are such assumptions, are they learnt from data (20 trials) rather than found by human trial (unlimited trials in fact, even if the environment changed, the underlying structure won't change much, so some rules may remain effective) ?

3 Acknowledgement

Jiaqi Yang is one of the most active competitor on the forum (id: thyrixyang), and the one that received most help from organizers and other competitors.

Many thanks to oetbent, super11 and sekou, for your timely response and patience in explaining the competition rule.

Thanks to tutorial authers: Oetbent and Ainilaha, your tutorials provided foundations and many inspirations of our algorithm design.

4 Requirement

The only requirement is scikit-learn and numpy, which are the most commonly used lib in scientific research (and provided by organizer as mentioned on forum).

My environment is much messier so the requirement.txt file may not be used.