Thompson sampling approach for Recommendation systems

Reinforcement Learning Project
Summer Term 2020

Victor Garcia - 414248
Supervisor: Maghsudi Setareh

Contents

1 Introduction

In many online sequential decision-making problems, such as contextual bandits [4] and reinforcement learning [17], an agent learns to take a sequence of actions to maximize its expected cumulative reward, while repeatedly interacting with an unknown environment. Moreover, since in such problems the agent's actions affect both its rewards and its observations, it faces the well-known exploration-exploitation trade-off. Consequently, the exploration strategy is crucial for a learning algorithm:

Under-exploration will typically yield a sub-optimal strategy, while over-exploration tends to incur a significant exploration cost. Various exploration strategies have been proposed, including epsilon-greedy (EG), Boltzmann exploration [16] upper-confidence-bound (UCB) [3] type exploration, and Thompson sampling (TS). Among them, TS [18], which is also known as posterior sampling or probability matching, is a widely used exploration strategy with good practical performance [5] and theoretical guarantees [13] [1].

Because the exact posterior is intractable, evaluating these approaches is hard. Furthermore, these methods are rarely compared on benchmarks that measure the quality of their estimates of uncertainty for downstream tasks. To address this challenge, we develop a benchmark for exploration using deep neural networks.

In this project, we investigate how the posterior approximations affect the performance of Thompson Sampling from an empirical standpoint. We test the performance in four different typical recommendation Datasets.

2 Contextual Bandit

2.1 Exploration technique

The contextual bandit problem works as follows. A_t time t = 1, . . . , n a new context X_t Rd arrives and is presented to algorithm A. The algorithm —based on its internal model and Xt— selects one of the k available actions, at. Some reward $r_t = r_t(X_t, a_t)$ is then generated and returned to the algorithm, that may update its internal model with the new data. At the end of the process, the reward for the algorithm is given by $r = \sum_{t=1}^{n} r_t$, and cumulative regret is defined as $R_A = E[r - r]$, where r is the cumulative reward of the optimal policy (i.e., the policy that always selects the action with highest expected reward given the context). The goal is to minimize R_A.

The main research question we address in this paper is how approximated model posteriors affect the performance of decision making via Thompson Sampling (Algorithm 1) in contextual bandits. We study a variety of algorithmic approaches to approximate a posterior distribution, together with different empirical and synthetic data problems that highlight several aspects of decision making. We consider distributions over the space of parameters that completely define a problem instance . For example, could encode the reward distributions of a set of arms in the multi-armed bandit scenario, or –more generally– all the parameters of an MDP in reinforcement learning.

Thompson Sampling is a classic algorithm [2] which requires only that one can sample from the posterior distribution over plausible problem instances (for example, values or rewards). At each round, it draws a sample and takes a greedy action under the optimal policy for the sample. The posterior distribution is then updated after the result of the action is observed.

Thompson Sampling has been shown to be extremely effective for bandit problems both in practice [5] and theory [1]. It is especially appealing for deep neural networks as one rarely has access to the full posterior but can often approximately sample from it.

In the following sections we rely on the idea that, if we had access to the actual posterior t given the observed data at all times t, then choosing actions using Thompson Sampling would lead to near-optimal cumulative regret or, more informally, to good performance. It is important to remark that in some problems this is not necessarily the case; for example, when actions that have no chance of being optimal still convey useful information about other actions. Thompson Sampling (or UCB approaches) would never select such actions, even if they are worth their cost [14].

In addition, Thompson Sampling does not take into account the time horizon where the process ends, and if known, exploration efforts should be tuned accordingly [15]. Nonetheless, under the assumption that very accurate posterior approximations lead to efficient decisions, the question is: what happens when the approximations are not so accurate? In some cases, the mismatch in posteriors may not hurt in terms of decision making, and we will still end up with good decisions. Unfortunately, in other cases, this mismatch together with its induced feedback loop will degenerate in a significant loss of performance. We would like to understand the main aspects that determine which way it goes. This is an important practical question as, in large and complex systems, computational sacrifices and statistical assumptions are made to favor simplicity and tractability. But, what is their impact?

2.2 Posterior Approximation Algorithm

We implement a Bayesian neural network by modeling each individual weight posterior as a univariate Gaussian distribution.
Thompson sampling then samples a network at each time step by sampling each weight independently. All weights in our neural networks are represented by probability distributions over possible values, rather than having a single fixed value as is the norm (see Figure 1). Learnt representations and computations must therefore be robust under perturbation of the weights, but the amount of perturbation each weight exhibits is also learnt in a way that coherently explains variability in the training data.

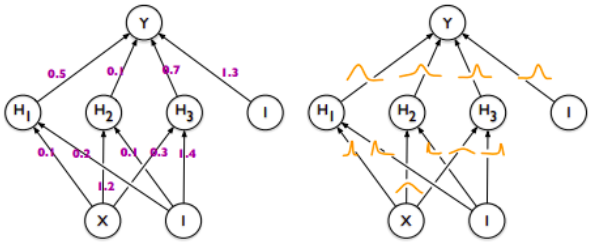

Figure 1: Left: each weight has a fixed value, as provided by classical backpropagation. Right: each weight is assigned a distribution, as provided by Bayes by Backprop.

In general, exact Bayesian inference on the weights of a neural network is intractable as the number of parameters is very large and the functional form of a neural network does not lend itself to exact integration. Instead we take a variational approximation to exact

Bayesian updates. We build upon the work of [6], who in turn built upon the work of [8]. In contrast to this previous work, we show how the gradients of [6] can be made unbiased and further how this method can be used with non-Gaussian priors. Consequently, Bayes by Backprop attains performance comparable to that of dropout [9]. Our method is related to recent methods in deep, generative modelling [10], where variational inference has been applied to stochastic hidden units of an autoencoder. Whilst the number of stochastic hidden units might be in the order of thousands, the number of weights in a neural network is easily two orders of magnitude larger, making the optimisation problem much larger scale. Uncertainty in the hidden units allows the expression of uncertainty about a particular observation, uncertainty in the weights is complementary in that it captures uncertainty about which neural network is appropriate, leading to regularisation of the weights and model averaging.

We use this uncertainty to drive exploration in our contextual bandit problem. Weights with greater uncertainty introduce more variability into the decisions made by the network, leading naturally to exploration. As more data are observed, the uncertainty can decrease, allowing the decisions made by the network to become more deterministic as the environment is better understood.

2.3 Data Transformation and Data Processing

The four datasets that we are using (Jester [19], EachMovies [12], MovieLens100k [7]and ModCloth [11]) have some differences in terms of how is the data presented. The Jester dataset is presented as a matrix which users as rows and items (jokes) as columns. However the other 3 datasets both Users and items are presented in columns. After processing this data we transform this to a matrix form.

The other big problem that we found with the data is the sparse matrix that we get after this transformation. For example in the case of the MovieLens or EachMovies Dataset, each user only rate some films from the 1000 available. For this reason, we get a matrix wich has almost all their positions as gaps.

The most straightforward solution that we found to avoid these gaps is to fill them with the average of the ratings from each user.
So for example, if one user has rated 250 films with the maximum rating (5) and 250 with the lowest rating (1), the other 500 films that are incomplete, will be rated with a 3. This will allow us to have much more data preserving the likes of the user. We evaluate this approach to really check whether this helps to preserve the likes of the user or not.

3 Results

The following results have been done mainly with the libraries Tensorflow, Pandas and NumPy.
We run the bandit 1000 rounds in each dataset. With the purpose of getting more reliable statistics, we repeat this process 10 times in each dataset, and then we take the average.
Since each dataset doesn't have the same optimal reward, the cumulative reward doesn't show us clearly in which dataset the bandit it's performing better. We compute the percentage of reward to clearly notify in which dataset the bandit is being able to learn.
For the context of the bandit we consider the 70% of the data from each user and for the arms, the 30%. So for example if we have 1000 items that each user is rating, 700 will be

used as context and then the bandit will recommend one of the 300 items left, and will get the reward of it.

	Users	Items	Cumulative Regret E [r* - r]	Reward percentage % (r/r*)
EachMovie	72.910	1.628	33,85 - (-5,121) = 38.971	-15%
MovieLens	1000	1700	1306,68 - (-5,68) = 1392.36	-0.384%
Jester	73.421	100	69.478,1 - 61.738,13 = 7739.98	88%
ModCloth	44.783	1.020	67,5 - 0,947 = 66.553	1.34%

Table 1: Comparison of "Bayes by Backpropagation" algorithm in different datasets.

4 Conclusions

As we can see in the above table our contextual bandit perform really good in the Jester Dataset. Getting a cumulative regret of 7739.98.
However in the case of the other 3 datasets, their performance is quite poor. They accumulate too much regret. In some iterations the bandit even gets negative reward. This makes that the regret increase drastically.
The reason of these results are mainly 2:
The first one is the number of items. In Jester dataset there are only 100 items, therefore the contextual bandit can take 70 items as context and 30 as arms (jokes to recommend). With only 30 arms is feasible that the bandit learn which arm is better to select in each case. Nevertheless, in the case of the other 3 dataset, the number of items is at least 10 times more: MovieLens (1700 Items), EachMovie (1628 Items) or ModCloth (1020 Item). So in these cases the bandit has to learn which one of the 300 arms is better to pull in only 1000 rounds. This task is almost impossible for the bandit since each arm can only be pulled 3 times. With only 3 pulls of the arm we cannot know the probability distribution of that bandit.
This does justice to the results obtained: In Jester dataset, the bandit can learn which joke to recommend, but not in the other 3 cases.

The second reason of the above results are the density of the matrices. While the Jester is a roughly dense matrix, the MovieLens, Eachmovie and ModCloth are really sparse matrices. Since most of the matrix positions are empty, the bandit cannot get information from them. We tried to solve this problem of incomplete positions, filling these gaps with the average rating of each user. With this we have extremely more data. The problem of this approach is that since most of the films have the same rating, doesn't help to decide the kind of films that the user want to watch.

We can see this more clear with an example:
Let's suppose that we have 500 terror films and only 100 action films. Our hypothetical user loves action films and therefore he rates 4 or 5 action films with an outstanding rating. The average of those ratings is also an outstanding rating, so that the 500 terror films will have that average. What our bandit see it's that the user has rated 500 terror films with a wonderful rating. Therefore the bandit will recommend to the user only terror films, when our user only want to keep watching action films.
This is the second reason why the bandit in these 3 datasets it's not working well enough.

5 Limitations and Future Work

As we have said in the previous section this approach has a clear limitation when the datasets has too many items and/or when they are really sparse.
A possible line of future would be to combine this approach with any kind of matrix reduction for solving this problem of sparse matrices. A challenging line of work would be an hybrid system between the typical matrix factorization approach and this contextual bandit approach.
Apart from this, a general line of future work would be to find new approaches that learn the data patterns with few data.

References

[1] Shipra Agrawal and Navin Goyal. "Analysis of thompson sampling for the multi-armed bandit problem". In: *Conference on learning theory*. 2012, pp. 39–1.

[2] Shipra Agrawal and Navin Goyal. "Thompson sampling for contextual bandits with linear payoffs". In: *International Conference on Machine Learning*. 2013, pp. 127–135.

[3] Peter Auer. "Using confidence bounds for exploitation-exploration trade-offs". In: *Journal of Machine Learning Research* 3.Nov (2002), pp. 397–422.

[4] Sébastien Bubeck and Nicolo Cesa-Bianchi. "Regret analysis of stochastic and nonstochastic multi-armed bandit problems". In: *arXiv preprint arXiv:1204.5721* (2012).

[5] Olivier Chapelle and Lihong Li. "An empirical evaluation of thompson sampling". In: *Advances in neural information processing systems*. 2011, pp. 2249–2257.

[6] Alex Graves. "Practical variational inference for neural networks". In: *Advances in neural information processing systems*. 2011, pp. 2348–2356.

[7] GroupLens. *MovieLens 100k Dataset*. 2000. URL: `https://grouplens.org/datasets/movielens/100k/`.

[8] Geoffrey E Hinton and Drew Van Camp. "Keeping the neural networks simple by minimizing the description length of the weights". In: *Proceedings of the sixth annual conference on Computational learning theory*. 1993, pp. 5–13.

[9] Geoffrey E Hinton et al. "Improving neural networks by preventing co-adaptation of feature detectors". In: *arXiv preprint arXiv:1207.0580* (2012).

[10] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[11] Julian McAuley. *ModCloth Dataset*. 2000. URL: `https://cseweb.ucsd.edu/~jmcauley/datasets.html#clothing_fit`.

[12] Paul McJones. *EachMovies Dataset*. 1997. URL: `http://www.cs.cmu.edu/~lebanon/IR-lab/data.html`.

[13] Daniel Russo and Benjamin Van Roy. "An information-theoretic analysis of Thompson sampling". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 2442–2471.

[14] Daniel Russo and Benjamin Van Roy. "Learning to optimize via information-directed sampling". In: *Advances in Neural Information Processing Systems*. 2014, pp. 1583–1591.

[15] Daniel Russo et al. "A tutorial on thompson sampling". In: *arXiv preprint arXiv:1707.02038* (2017).

[16] Richard S Sutton. "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming". In: *Machine learning proceedings 1990*. Elsevier, 1990, pp. 216–224.

[17] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning. Vol. 135*. 1998.

[18] William R Thompson. "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples". In: *Biometrika* 25.3/4 (1933), pp. 285–294.

[19] Berkeley University. *Jester 100 jokes dataset*. 2000. URL: `http://eigentaste.berkeley.edu/dataset/`.