Multi-Armed Recommender System Bandit Ensembles

Rocío Cañamares
Universidad Autónoma de Madrid
rocio.cannamares@uam.es

Marcos Redondo
Universidad Autónoma de Madrid
marcos.redondo@estudiante.uam.es

Pablo Castells
Universidad Autónoma de Madrid
pablo.castells@uam.es

ABSTRACT

It has long been found that well-configured recommender system ensembles can achieve better effectiveness than the combined systems separately. Sophisticated approaches have been developed to automatically optimize the ensembles' configuration to maximize their performance gains. However most work in this area has targeted simplified scenarios where algorithms are tested and compared on a single non-interactive run. In this paper we consider a more realistic perspective bearing in mind the cyclic nature of the recommendation task, where a large part of the system's input is collected from the reaction of users to the recommendations they are delivered. The cyclic process provides the opportunity for ensembles to observe and learn about the effectiveness of the combined algorithms, and improve the ensemble configuration progressively.

In this paper we explore the adaptation of a multi-armed bandit approach to achieve this, by representing the combined systems as arms, and the ensemble as a bandit that at each step selects an arm to produce the next round of recommendations. We report experiments showing the effectiveness of this approach compared to ensembles that lack the iterative perspective. Along the way, we find illustrative pitfall examples that can result from common, single-shot offline evaluation setups.

CCS CONCEPTS

• Information systems → Recommender systems • Computing methodologies → Ensemble methods; Reinforcement learning.

KEYWORDS

Multi-armed bandits; Ensembles; Hybrid recommender systems; Interactive recommendation; Feedback loop.

ACM Reference format:

Rocío Cañamares, Marcos Redondo and Pablo Castells. 2019. Multi-Armed Recommender System Bandit Ensembles. In *Proceedings of the ACM Conference on Recommender Systems (RecSys'19). ACM, New York, NY, USA, 5 pages.*

1 Introduction

In the pursuit of maximal recommendation effectiveness it was soon realized in the field that the best results in recommendation are obtained by aggregating several algorithms into ensembles [4]. Besides a practical means for scrapping accuracy improvements, ensembles are a particular way to build hybrid systems that combine the strengths (and compensate the limitations) of different recommendation approaches, such as content-based and collaborative filtering methods [6,15]. Hybrid recommendation is routinely listed as a category on its own in recommender systems introductions and surveys [1].

One of the challenges in building ensembles lies in properly tuning the contribution of the combined algorithms to the aggregated output. Sophisticated approaches have been developed to automatically optimize the ensembles' configuration to maximize their performance gains. Ensembles should include individual algorithms with proven performance in order for the combination to be effective. However most work in this area has targeted simplified scenarios where algorithms are tested and compared on a single non-interactive run where each user is delivered just one set of recommended items, on which a final evaluation metric is computed and the experiment ends. As runtime conditions (data, users, item catalog, etc.) evolve, the performance of previously well-behaved algorithms may degrade and hamper the ensemble effectiveness –or weaknesses may simply surface that had gone unnoticed in the selection phase. We may therefore wish that the participation of the combined algorithms be dynamically updated and readjusted to better reflect the latest accounts of their performance –and/or our current knowledge thereof– as they deliver partial outputs in a production system.

In this paper we consider a more realistic perspective beyond single-shot recommendations, bearing in mind the cyclic nature of the recommendation task, where a large part of the system's input is collected from the reaction of users to the recommendations (the system's output) they are delivered. The cyclic process provides the opportunity for ensembles to test, observe and learn about the effectiveness of the combined systems, and improve the ensemble configuration progressively, casting ensemble configuration as a reinforcement learning task [20,23]. In this perspective, we explore the adaptation of a multi-armed bandit approach to dynamically optimize recommender system ensembles, by representing the combined systems as arms, and the ensemble as a bandit that at each step selects an arm to produce the next round of recommendations.[1]

We adapt two basic bandit algorithms –Thompson sampling [8] and ε-greedy [23]– and verify that the resulting approaches are empirically more effective than alternative ensemble techniques that lack the long-term perspective in experiments based on offline datasets. Along the way, we find illustrative pitfall examples such as overfitting behavior, self-defeating reinforcement loops, and poor decisions that can result from common, single-shot offline evaluation setups.

RecSys'19, September 16–20, 2019, Copenhagen, Denmark.

ACM ISBN 978-1-4503-6243-6/19/09…$15.00 https://doi.org/10.1145/3298689.3346984

[1] An implementation of all the methods and experimental procedures described in this paper is publicly available at https://github.com/ir-uam/EnsembleBandits.

2 Related Work

2.1 Recommender System Ensembles

Hybrid systems have been found to be good means to improve the performance separately obtained by effective individual recommendation algorithms. Ensembles are a particular case of hybrid approach which regards the algorithms to combine as black boxes. This has the powerful advantage of allowing the combination of as many algorithms as we wish, and of any type [4,6,23]. The vast majority of research in this area has focused on running recommendations only once. There are, however, some studies that seek to dynamically improve the ensembles [2,3,9,15]. An important issue in most such prior work is the use of learning and optimization techniques (bagging, boosting, fusion, randomness injection) to compute the best parameter configuration of complex models. This can potentially increase the execution time of the ensemble, hindering its scalability in production scenarios. Moreover, all of them imply the execution, at least once, of all the combined algorithms in order to identify the best one.

2.2 Bandit Recommender Systems

The application of multi-armed bandit techniques has started to become popular in the recommender systems field. Work in this area has used bandit techniques to select the next item to recommend by considering all the candidate items as arms of the bandit [14,16,17,25,26]. There is barely any precedent in modeling ensemble components as arms, as we explore in this paper. Closest to our present research is the work by Brodén et al. [5], though it is developed in non-personalized item-to-item recommendation in a quite specific e-commerce context. Pang et al. [20] also propose to use bandits as ensembles of algorithms, but not for recommender systems. And other work that relates bandits and ensembles has focused in combining algorithms that are bandits, that is, in creating ensembles of bandits [24].

One important application of bandit ensembles is A/B testing [12,22], where a bandit method automatically decides between several algorithms based on their previous performance. If one recommendation algorithm is clearly less effective than the others, the bandit ensemble will progressively reduce the traffic it is assigned. In this line, we may say that bandit ensembles have been already used for A/B testing to select a winner between several candidates [11] –such winner is simply the algorithm that has been selected more times by the ensemble. Bandit A/B testing implicitly assumes that some of the tested systems will be called increasingly less often by the bandit, or that we are constrained (by whatever environmental conditions) to making an excluding choice among the alternatives. But this might not be the case: there may not be a clear winner, or the reward may not be stationary and the best algorithm in the past is not necessarily so in the future. And the A/B test traffic may not monotonically decrease towards zero for all systems but one sought champion. And we may not have any major deterrent in leaving the ensemble running indefinitely if it happens to be more effective itself than either of the tested variants. Such conditions motivate the perspective we explore here.

3 Bandit Recommender System Ensembles

We propose to adapt the multi-armed bandit approach for building recommender system ensembles. Our proposed bandit formulation of recommender ensembles is as follows:

- The **context** is the target user to whom a recommendation is to be delivered.
- The **arms** are the recommendation algorithms that are combined in the ensemble. When an arm is selected, the corresponding algorithm is run to select one item to be recommended to the target user.
- The **reward** is 1 if the user is pleased by the recommended item, and 0 otherwise.
- Arms can be **updated** after each individual recommendations, or every certain number (a batch) of recommendations [21]. In our experiments, we will select target users in a loop over all users, and update the arms after each such round.

Following the multi-armed approach, the selection of the algorithm to recommend at each step is based on its performance in the previous cycles in which it has been selected. For instance, in ε-greedy [23], the algorithm (arm) that has the highest average reward (i.e. the precision) so far is selected with probability $1 - \varepsilon$, and with probability ε an algorithm is selected uniformly at random regardless of its historical effectiveness. For Thompson sampling [8], the posterior of the unknown reward distribution of each arm is modeled as a Beta(α_a, β_a) distribution, where α_a and β_a are the number of successful and unsuccessful recommendations of algorithm a, respectively. A value p_a is drawn for each arm from its Beta distribution, and the arm with highest value is selected.

Note that the previous bandit reformulation can be employed to transform any multi-armed bandit solution into a recommender ensemble. In the next section we specifically chose to implement the previous explained popular bandits (Thompson sampling and ε-greedy) to verify our proposal, but any other bandit method could be used following the previous approach.

4 Experiments

4.1 Data

In order to verify the performance of the bandit recommender ensembles that we have proposed in the previous section, we run the algorithms using data from the MovieLens 1M dataset [18], containing 1,000,209 ratings by 6,040 users to 3,706 movies. We binarize the ratings by mapping values 1-3 to 0 and 4-5 to 1.

4.2 Algorithms

In order to be able to closely examine the behavior of our bandit ensembles, we shall test the combination of just three recommendation algorithms: two well-performing collaborative filtering methods (kNN [19] and matrix factorization [13]), and non-personalized most-popular recommendation, which represents a good option when the data is too sparse to obtain any reliable personalized signal. As a baseline ensemble we implement an alternative dynamic ensemble, which we apply to the same three algorithms. At each point in time, this ensemble applies an offline evaluation of the combined algorithms –by randomly splitting the input data collected so far into training and test subsets– and chooses the best one (in Precision@1) to produce the next round of recommendations –now taking all the available data as input training. We will also compare our approach, as a point of reference, to the individual algorithms combined in the ensembles, as well as random recommendation.

We tune the configuration of our bandit ensembles by grid search. Thus, we initialize the Thompson sampling with $\alpha_a =$ 1,000 and $\beta_a = 1$ for all the three combined recommendation algorithms, and ε = 0.1 for the ε-greedy bandit. We configure kNN with cosine similarity and all users as neighbors, for simplicity; and we take the configuration of matrix factorization reported in [7] for MovieLens 1M: $\alpha = 1$, $\lambda = 0.1$, $k = 20$, and 20 iterations.

4.3 Offline Evaluation Approach

Our evaluation approach simulates an environment where users discover and rate the items that recommender systems iteratively suggest them, providing new feedback that the systems can incorporate as input in the next round.

We start from an initial training set including 5% of the available ratings sampled uniformly at random (equivalent to ~10 ratings per user). We run the evaluated recommender system (ensemble or standalone algorithms) with the training set as input, and use the remaining 95% of the data as the test set to simulate user feedback. At each round (epoch) of the simulation, the system recommends one item to each user (bandits pulling an arm per user); for each user, the resulting reward is 1 if a positive rating is available in the test set for the recommended item, and 0 otherwise. If a (positive or negative) rating was available, it is added to the training set, and the cycle goes on. If not, we keep track and take care that the same item is not recommended again to the same user. Note that each recommender system has its own training, test and exclusion sets, since they are built from the user feedback to their own specific recommendations. Thus, only the initial training set is common to all the compared systems. The algorithms within ensembles do share the same data, collected by the ensemble.

For Thompson sampling, the value of the parameters α_a and β_a corresponding to each algorithm a (popularity, kNN and matrix factorization) is updated after each epoch, by incrementing them in the number of, respectively, hits and misses that a has obtained in the epoch. Since one epoch implies recommending an item for each user, the number of hits and misses can be in the order of thousands. For this reason we initialize α_a as 1,000, as a highly optimistic initialization promoting exploration in the early stages, in order to avoid the bandit getting stuck with the algorithm that happens to obtain more hits in the first few epochs. Analogously, the ε-greedy bandit updates the hit rate of each candidate algorithm after each epoch.

4.4 Results

Figure 1 shows the results for the first 200 epochs. The number of epochs corresponds to the number of recommended items suggested to each user, since the recommender systems only recommend one item per epoch in our setting. Thus, we consider that 200 recommendations per user is a reasonable snapshot point to compare the different approaches. The performance of each recommender system (ensembles and standalone algorithms) is computed as the cumulative recall achieved up to each epoch. That is, as the number of successful recommendations, divided by the total number of relevant ratings in the initial test set.

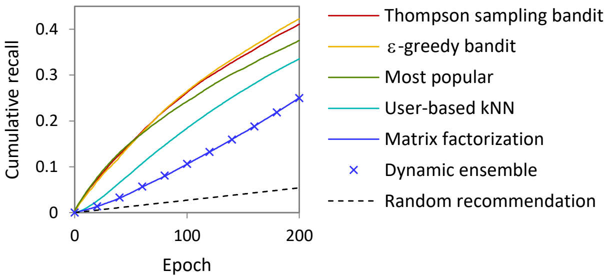

Figure 1: Incremental recall (y axis) vs. number of recommendations (x axis) for the tested recommendation algorithms, including the bandit ensembles.

We can see that the bandit ensembles clearly outperform all other alternatives, which in fact work below what one might expect due to their myopic nature, targeting optimal one-shot recommendations. ε-greedy seems to do slightly better than Thompson sampling.

The collaborative filtering algorithms are not able to do better than recommendation by popularity. This can be attributed to their vulnerability to the initial data sparsity, while non-personalized popularity gets an advantage in the first few iterations, when there is not enough data for exploitation-oriented collaborative filtering to produce reliable personalized recommendations. After the first 50 recommendations we can see that kNN and matrix factorization start to catch up (their slope is steeper) but it is not enough for outperforming popularity.

The poor performance of the non-bandit ensemble may come at some surprise. The ensemble appears to select matrix factorization all the time, which seems to be a bad choice: the matrix factorization algorithm is optimal for a single recommendation, but shows to be quite suboptimal as a cyclic recommendation approach. An additional reason for the ensemble to stick to this selection can be attributed to a feedback loop effect [10]: since the ensemble is building its own training set from the recommendations of matrix factorization, an offline evaluation (used by the ensemble to select a winner) using such training data (via a random split) is biased in favor of the algorithm that collected the ratings. We will further analyze this phenomenon in the next section, showing how the ensemble is strongly dependent on the winner of the first iteration.

In order to better understand the behavior of our bandit ensembles, Figure 2 shows the fraction of users for whom each of the three combined algorithms has been selected by the ensemble to produce the recommendation at each epoch. We can see that popularity is clearly dominant in the first iterations, but as the matrix factorization starts to improve, the bandit ensembles gradually increase the selection of the latter. kNN, however, seems to be rarely selected by the ensembles even though, as a standalone iterative recommender, it is better than matrix factorization. This comes to show that bandit ensembles are able to achieve non-obvious enhanced optimizations.

The behavior observed in Figure 2 also illustrates the fact that keeping more than one algorithm, at the appropriate traffic ratio,

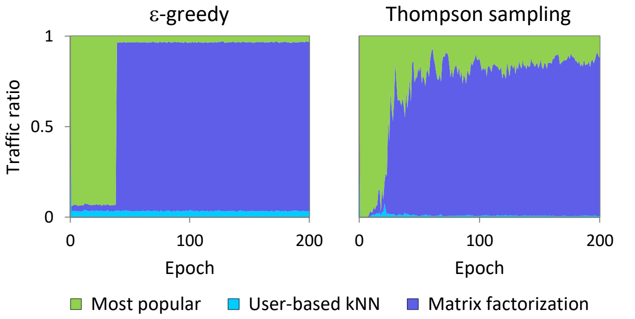

Figure 2: Fraction of users that the ε-greedy bandit (left) and the Thompson sampling bandit (right) have selected to be recommended by each of the basic algorithms that compose the bandit.

can be better than choosing only one. This justifies the employment of bandit ensembles as alternative to conventional test A/B experiments, leaving thus the selection of the candidates in the hands of the bandit ensemble. If there is a clear winner, the bandit will reduce the fraction of users for whom other options are selected to a minimum, as happens with kNN in our experiments.

We also see that even though our two bandits soon favor matrix factorization as a preferred algorithm, the initial reliance on popularity, albeit short, seems to not only directly improve the ensemble, but also considerably (indirectly) enhance the behavior of matrix factorization itself thereafter (by collecting a richer pool of user feedback as input data), compared to an exclusive and persistent reliance on matrix factorization alone, as does the dynamic ensemble. The continued (albeit small) fraction of alternation among algorithms, can also help in the same fashion.

4.5 Overfitting and Reinforced Feedback Loop

The tendency of the baseline ensemble to select matrix factorization over the two other alternatives can be explained because matrix factorization is likely very good, in an offline evaluation, at (over)fitting whatever data has been collected. To provide further insights on the feedback loop effects in the baseline ensemble, we run additional experiments where we reduce the initial split ratio to 1% data for training, thus creating an even colder start with just 1 or 2 ratings per user in the initial training set. In such conditions, the variance of the initial random selection of the training set produces different winners in the dynamic non-bandit ensemble at the first epoch, so that matrix factorization is not always the selected algorithm in the first iteration.

Figure 3 shows different executions of the baseline ensemble and the basic algorithms with different random samples of the initial data. We zoom into the first 50 epochs to see the initial behavior of the ensemble that determines its subsequent evolution. While popularity, kNN, and matrix factorization behave quite the same in the different executions, the baseline ensemble switches between them in different instances of the same experiment. In most of the cases, the algorithm selected in the first iteration is kept thereafter by the ensemble, confirming the feedback loop effect. If the winner happens to be kNN or matrix factorization, the ensemble shall underperform compared to popularity.

There are −less frequent− situations, however, when the winner changes along the cycles, as is the case in the bottom-right graph of Figure 3. This only happens in the first few iterations though, and the winner stabilizes after epoch 5. Note, on the other hand, that in no situation the ensemble outperforms the best standalone algorithm (popularity).

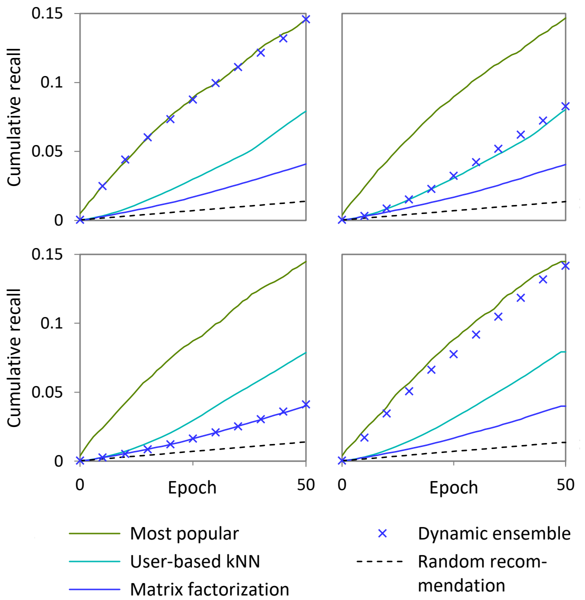

Figure 3: Incremental recall (*y* axis) vs. number of recommendations (*x* axis) for the baseline ensemble and the algorithms it combines. Each graph corresponds to one different execution of the offline experiment.

One might consider that this dynamic ensemble is just a bad idea. Note however that it represents the way an algorithm would be selected in a typical manual offline evaluation methodology, as has been widespread practice in the field to date. In contrast, the bandit ensembles seem not to suffer from this feedback loop bias, and are able to obtain better results than any of the combined algorithms achieve separately.

5 Conclusions

We have explored the incorporation of multi-armed bandit techniques to the design of dynamic recommender ensembles that select the best among several algorithms based on the previous performance of each candidate. We have shown that the bandit approach is empirically effective, improving not only the individual algorithms, but also other ensemble alternatives. In particular, the bandit approach does not suffer from the feedback loop bias that is evidenced in decisions based on offline evaluation with logged data.

An additional important advantage of bandit ensembles is their low computational cost: they need to run just one selected recommendation algorithm, and not all the others. The savings can be considerable when the combined algorithms are computationally involved, and the ensemble size is large [4].

ACKNOWLEDGMENTS

This work was partially supported by the Spanish Government (grant nr. TIN2016-80630-P).

REFERENCES

[1] G. Adomavicius and A. Tuzhilin (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17, 6 (June 2005), 734–749.

[2] F. Aksel and A. Birtürk (2010). An Adaptive Hybrid Recommender System that Learns Domain Dynamics. In *International Workshop on Handling Concept Drift in Adaptive Information Systems: Importance, Challenges and Solutions* (HaCDAIS-2010) *at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases* (ECML PKDD 2010). Barcelona, Spain, 49–56.

[3] A. Bar, L. Rokach, G. Shani, B. Shapira and A. Schclar (2013). Improving Simple Collaborative Filtering Models Using Ensemble Methods. In *11th International Workshop on Multiple Classifier Systems* (MCS 2013). Nanjing, China, 1–12.

[4] R. Bell, Y. Koren and C. Volinsky (2007). The bellkor solution to the Netflix prize. *KorBell Team's Report to Netflix* (2007).

[5] B. Brodén, M. Hammar, B. Nilson and D. Paraschakis (2018). Ensemble Recommendations via Thompson Sampling: an Experimental Study within e-Commerce. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces* (IUI 2018). Tokyo, Japan, 19–29.

[6] R.Burke (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12, 4 (November 2002). Kluwer Academic Publishers Hingham, MA, USA, 331–370.

[7] R. Cañamares and P. Castells (2017). A Probabilistic Reformulation of Memory-Based Collaborative Filtering – Implications on Popularity Biases. In *Proceeding of the 40th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR 2017). ACM, New York, USA, 215–224.

[8] O. Chapelle and L. Li (2011). An empirical evaluation of Thompson Sampling. In *Proceedings of Neural Information Processing Systems* (NIPS 2011). Curran Associates, Inc., Red Hook, NY, USA, 2249–2257.

[9] S. Dooms (2013). Dynamic generation of personalized hybrid recommender systems. In *Proceedings of the 7th ACM Conference on Recommender Systems* (RecSys 2013). Hong Kong, China, 443–446.

[10] A. Gilotte, C. Calauzènes, T. Nedelec, A. Abraham and S. Dollé (2018). Offline A/B Testing for Recommender Systems. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining* (WSDM 2018). ACM, New York, NY, USA, 198–206.

[11] D. Hill, H. Nassif, Y. Liu, A. Iyer and S. Vishwanathan (2017). An Efficient Bandit Algorithm for Realtime Multivariate Optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD 2017). Halifax, NS, Canada, 1813–1821.

[12] R. Kohavi, R. Longbotham, D. Sommerfield and R. Henne (2009). Controlled experiments on the web: survey and practical guide. *Data Mining and Knowledge Discovery*, 18, 1 (February 2009), 140–181.

[13] Y. Hu, Y. Koren and C. Volinsky (2008). Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining* (ICDM 2008). IEEE Computer Society, Washington, DC, USA, 15–19.

[14] J. Kawale, H. H. Bui, B. Kveton, L. Tran-Thanh and S. Chawla (2015). Efficient Thompson Sampling for Online Matrix-Factorization Recommendation. In *Proceedings of Neural Information Processing Systems* (NIPS 2015). Curran Associates, Inc., Red Hook, NY, USA, 1297–1305.

[15] P. Kouki, S. Fakhraei, J. Foulds, M. Eirinaki and L. Getoor (2015). HyPER: A Flexible and Extensible Probabilistic Framework for Hybrid Recommender Systems. In *Proceedings of the 9th ACM Conference on Recommender Systems* (RecSys 2015). ACM, New York, NY, USA, 99–106.

[16] L. Li, W. Chu, J. Langford and R. Schapire (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web* (WWW 2010). ACM, New York, NY, USA, 661–670.

[17] S. Li, A. Karatzoglou, and C. Gentile (2016). Collaborative Filtering Bandits. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR 2016). ACM New York, NY, USA, 539–548.

[18] F. M. Maxwell and J. A. Konstan (2015). The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems*, 5, 4 (December 2015).

[19] X. Ning, C. Desrosiers and G. Karypis (2015). A Comprehensive Survey of Neighborhood-Based Recommendation Methods. In F. Ricci, L. Rokach and B. Shapira (Eds.), *Recommender Systems Handbook* (2nd ed.). Springer, New York, NY, USA, 37–76.

[20] K. Pang, M. Dong, Y. Wu and T. Hospedales (2018). Dynamic Ensemble Active Learning: A Non-Stationary Bandit with Expert Advice. *In Proceedings of the 24th International Conference on Pattern Recognition* (ICPR 2018). IEEE Computer Society, Washington, DC, USA, 2269–2276.

[21] V. Perchet, P. Rigollet, S, Chassang and E. Snowberg (2016). Batched Bandit Problems. *Annals of Statistics*, 44, 2 (April 2016), 660–681.

[22] D. Siroker and P. Koomen (2015). *A/B testing: the most powerful way to turn clicks into customers*. John Wiley & Sons Inc, Hoboken, NJ, USA, 2015.

[23] R. Sutton and A. Barto (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press, Cambridge, MA, USA, 2018.

[24] L. Tang, Y. Jiang, L. Li and T. Li (2014). Ensemble contextual bandits for personalized recommendation. In *Proceedings of the 8th ACM Conference on Recommender Systems* (RecSys 2014). Foster City, CA, USA, 73–80.

[25] Q. Wang, C. Zeng, W. Zhou, T. Li, S. S. Iyengar, L. Shwartz and G. Grabarnik (2019). Online Interactive Collaborative Filtering Using Multi-Armed Bandit with Dependent Arms. *IEEE Transactions on Knowledge and Data Engineering*, 31, 8 (August 2019), 1569–1580.

[26] X. Zhao, W. Zhang and J. Wang (2013). Interactive Collaborative Filtering. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management* (CIKM 2013). ACM, New York, NY, USA, 1411–1420.