Deep density networks and uncertainty in recommender systems

Yoel Zeldes
Taboola
Tel Aviv, Israel
yoel.z@taboola.com

Stavros Theodorakis
DeepLab
Athens, Greece
sth@deeplab.ai

Efrat Solodnik
Taboola
Tel Aviv, Israel
efrat.s@taboola.com

Aviv Rotman
Taboola
Tel Aviv, Israel
aviv.r@taboola.com

Gil Chamiel
Taboola
Tel Aviv, Israel
gil.c@taboola.com

Dan Friedman
Taboola
Tel Aviv, Israel
dan.f@taboola.com

ABSTRACT

Building robust online content recommendation systems requires learning complex interactions between user preferences and content features. The field has evolved rapidly in recent years from traditional multi-arm bandit and collaborative filtering techniques, with new methods employing Deep Learning models to capture non-linearities. Despite progress, the dynamic nature of online recommendations still poses great challenges, such as finding the delicate balance between exploration and exploitation. In this paper we show how uncertainty estimations can be incorporated by employing them in an optimistic exploitation/exploration strategy for more efficient exploration of new recommendations. We provide a novel hybrid deep neural network model, Deep Density Networks (DDN), which integrates content-based deep learning models with a collaborative scheme that is able to robustly model and estimate uncertainty. Finally, we present online and offline results after incorporating DNN into a real world content recommendation system that serves billions of recommendations per day, and show the benefit of using DDN in practice.

KEYWORDS

Recommendation Systems, Deep learning, Uncertainty, Mixture Density Networks

1 INTRODUCTION

In order to navigate the vast amounts of content on the internet, users either rely on search queries, or on content recommendations
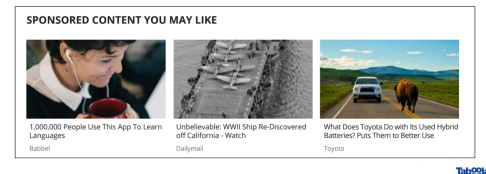
Figure 1: Taboola's recommendation widget example.

powered by algorithms. Taboola's content discovery platform leverages computational models to match content to users who are likely to engage with it. Taboola's content recommendations are shown in widgets that are usually placed at the bottom of articles (see Fig. 1) in various websites across the internet, and serve billions of recommendations per day, with a user base of hundreds of millions of active users.

Modeling in recommendation systems can be classified into either Collaborative Filtering (CF) or content-based methods. CF methods use past user-item interactions to predict future ratings [11] usually realized by Matrix Factorization (MF) [12]. A drawback to MF approaches is the cold-start (CS) problem. Content-based approaches mitigate CS by modeling explicitly meta-information about the items. Hybrid methods that combine both the memorization and generalization advantages have also been proposed [5]. We use this kind of hybrid approach, by employing deep neural networks (DNNs) to learn item representations and combining those with contextual features.

In order to improve long-term performance and tackle faster the CS problem, recommender systems have been modeled in a multi-arm bandit setting, where the goal is to find an *exploitation* and *exploration* selection strategy that maximizes the long term reward [10]. One of the basic approaches to deal with multi-arm bandit problems is the ϵ-greedy algorithm. Upper Confidence Bound (UCB) [1] and Thompson sampling techniques [16] use uncertainty estimations in order to perform more efficient exploration of the feature space, either by explicitly adding the uncertainty to the estimation or by sampling from the posterior distribution respectively. Estimating uncertainty is crucial in order to utilize these methods. To deal with this, bayesian neural networks [14] using distributions over the weights were applied by using either sampling or stochastic variational inference [9, 15]. [3] proposed Bayes
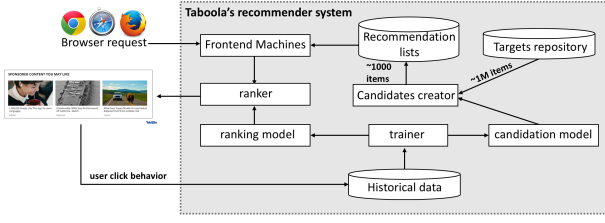
Figure 2: High level overview of candidation and ranking architecture.

by Backprop algorithm for the variational posterior estimation and applied Thompson sampling in a multi-arm bandit setting similarly to our case. [6] proposed Monte Carlo (MC) dropout, a Bayesian approximation of model uncertainty achieved by extracting estimations from the different sub-models that have been trained using dropout. Building upon their previous work, the authors separated uncertainty into two types, model and data uncertainty, while studying the effect of each uncertainty separately in computer vision tasks [8]. Similarly, we separate recommendation prediction uncertainty into three types: measurement, data and model uncertainty. In contrast to [17], we assumed heteroscedastic data uncertainty which was a more natural choice for recommendation systems. Our work has parallels to [10] where the authors formulated the exploration/exploitation trade-off in personalized article recommendation as a contextual bandit problem proposing LinUCB which adapts the UCB strategy. Our approach extends LinUCB by using a deep model instead, while explicitly modeling and estimating uncertainty. Finally, inspired by [7, 13] we model measurement noise using a Gaussian model and combine it with a Gaussian Mixture Model (GMM) to form a deep Mixture density network (MDN) [2].

In this paper we introduce a unified hybrid DNN to explicitly model and estimate measurement, data and model uncertainty and utilize them to form an optimistic exploitation/exploration selection strategy that is applied in a real world and large-scale content recommendation system. We explicitly model recommendations' content and combine it with context by using a collaborative fusion scheme. To the best of our knowledge this is the first time that a hybrid DNN model with uncertainty estimations is employed in a multi-arm bandit setting for recommender systems.

2 TABOOLA'S RECOMMENDER SYSTEM OVERVIEW

Taboola's revenue stream is facilitated by online advertisers, who pay a fixed amount CPC for each click while the total value is measured in $RPM = CTR * CPC * 1000$ and CTR is the click probability of a recommendation. Taboola's challenge is to estimate CTR in any given context. Taboola's recommendation engine needs to provide recommendations within strict time constraints ($< 50ms$). As it is infeasible to rank millions of recommendations in that time frame, we have partitioned the system into *candidation* and *ranking* Fig. 2. During the candidation, we narrow down the list of possible recommendations. When we get request for recommendations, they retrieve the relevant ready-made recommendation list, and perform

an additional ranking of the recommendations based on additional user features using a DNN, further personalizing recommendations. This system architecture shows similarities to ([5]).

Due to the dynamic nature of Taboola's marketplace our algorithm needs to evaluate new recommendations, with tens of thousands of new possible recommendations every day. To support this, we split the algorithm into *exploration* and *exploitation* modules. Exploitation aims to choose the recommendations maximizing RPM, while exploration aims to enrich the dataset. In this paper we focus on the candidation phase and the corresponding CTR prediction task, leaving out of the scope the second ranking step.

3 DEEP DENSITY NETWORK

Our deep recommender model is a hybrid content-based and collaborative filtering (CF) system (Fig. 3). We use two DNN subnets to model target and context features. The target subnet gets as input the content features seen by user together with additional categorical features which are unseen to the user. The categorical features are passed through an embedding layer and concatenated with the content features, followed by fully-connected layers with a RELU activation function, resulting in the target feature descriptor. Similarly, the context features are modeled using a DNN, taking as input context features such as device type where the target is recommended, resulting in the context feature descriptor. The target and context feature descriptors are then fused in a collaborative filtering manner and finally passed through a fully-connected layer which outputs the parameters of a GMM i.e. (α_i, μ_i and σ_i). This GMM model is employed in order to model data uncertainty as discussed in sec. 4.1. In addition a Gaussian model is employed taking as input measurement uncertainty σ_ϵ in order to deconvolve data and measurement uncertainty as discussed in sec. 4.2.

In order to train our models, we use historical data which consists of target and context pairs (t, c), where t is the target we recommended in a specific browsing context c accompanied with a binary variable which indicates if the recommendation was clicked by the user. A natural choice would be to estimate CTR using a logistic loss. However, our data contains great variability in terms of CTR due to various factors which are external to the content itself; As an example, a widget which contains very large images will be more likely to capture the user's attention, which subsequently increases the probability of a click. Moreover, even inside a certain widget, the specific location of a recommendation (top left, bottom right) can have a vast impact on the eventual CTR, which is independent of the content itself. To account for this, building upon our previous work [4], we use a calibrated version of the CTR, to diminish the variability due to different contexts in which a recommendation was shown. In practice we train our DNN network to predict the log of the calibrated CTR using Maximum Likelihood Estimation (MLE), as this allow us to estimate unconstrained scalar values roughly normally distributed with zero-mean. From hereafter we will refer to log calibrated CTR simply as CTR.

4 UNCERTAINTY IN RECOMMENDER SYSTEMS

We separate uncertainty into three different types: data, measurement, and model uncertainties, and study the role of each one in
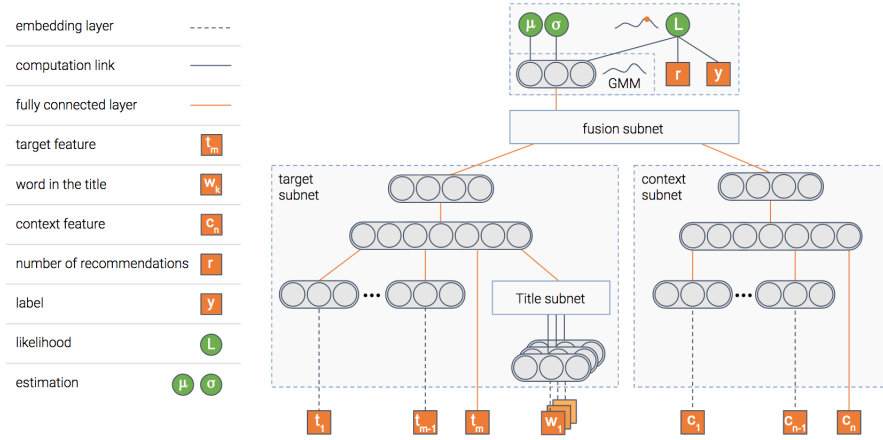
Figure 3: A hybrid content-based and collaborative filtering model accounting for data and measurement uncertainties. Both target and context features are passed through a DNN, then through a fusion sub-network, and finally through a fully-connected layer that outputs the parameters of a GMM. The number of recommendations r is used for attenuating the loss of noisy examples.

recommender systems. In addition, we provide a deep unified framework for explicitly modeling and estimating all types and further exploit them to form an optimistic exploration strategy (sec. 4.4).

4.1 Data Uncertainty

Data uncertainty corresponds to the inherent noise of the observations; it cannot be reduced even if more data was to be collected and is categorized into homoscedastic and heteroscedastic. Homoscedastic is constant over all different inputs, and heteroscedastic depends on the input, i.e. different input values may have more noisy outputs than others. A common source of data uncertainty in recommender system is temporal variability, wherein the CTR of the same content will change over time. This variance is an inherent property of the content, and changes with the type of the content; for instance a trending fashion product will have large temporal variance. An optimistic exploration strategy (sec. 4.4) can exploit estimations of this variability by prioritizing content that has larger variance as it *might* be trending at any given moment. This extends to non-temporal random variables that affect CTR and cannot be directly modeled.

We model data uncertainty by placing a distribution over the output of the model and learning it as a function of the input. To support that, we use a GMM with parameters (α_i, μ_i and σ_i) to model our observation: Y:

$$Y \sim \sum_i \alpha_i \mathcal{N}(\mu_i, \sigma_i^2) \tag{1}$$

4.2 Measurement Uncertainty

Measurement uncertainty corresponds to the uncertainty of the observed CTR due to the measurement noise introduced by the binomial recommendation experiment. This type of uncertainty depends on the number of times r a specific target $x = (t, c)$ pair was recommended, i.e. target t was recommended in context c.

In the previous section we saw that using a simple MDN model can estimate data uncertainty by employing a GMM at the last layer of our network. However, since the observed CTR is affected by the measurement noise, the estimated data uncertainty encapsulates also measurement uncertainty. In contrast, using the DDN model we explicitly model measurement noise and thus we are able to deconvolve measurement from data uncertainty.

Let Y, Y^* and ϵ be three random variables given $X = (t, c)$. Y corresponds to observed CTR, after recommending (t, c) pair, r times. Y^* corresponds to the true/clean CTR without the measurement noise, i.e. the CTR if we had recommended t infinite times in c. ϵ corresponds to the binomial noise error distribution.

$$Y = Y^* + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2), \quad Y^* \sim \sum_i \alpha_i \mathcal{N}(\mu_i, \sigma_i^2) \tag{2}$$

We approximate the binomial distribution of the measurement noise ϵ via a Gaussian model and model Y^* with a GMM. For every $Y^*|X$ we enforce constant $\sigma_\epsilon = f(\mu, r)$, where μ is the expected value of $Y^*|X$. This way, $Y^* \perp\!\!\!\perp \epsilon$ given X, as σ_ϵ depends only on r and μ. We can rewrite eq. 2 and deconvolve data and measurement uncertainties.

$$Y \sim \sum_i \alpha_i \mathcal{N}(\mu_i, \sigma_i^2 + \sigma_\epsilon^2) \tag{3}$$

To this end, the DDN model described in sec. 3 takes σ_ϵ as additional input and uses a Gaussian model to account for measurement uncertainty and predicts GMM's coefficients (α_i, μ_i and σ_i), from which we estimate the expected value and the standard deviation of Y^*.

4.3 Model Uncertainty

Model uncertainty accounts for uncertainty in the model parameters. This corresponds to the ignorance of the model and depends on the data that the model was trained on. For example, if a recommendation system chooses to show mainly sports articles, future training datasets will contain mostly sports articles. As a result,

the next trained model will have high model uncertainty for entertainment articles due to the lack of related content in the training dataset. This *feedback loop* is common in recommendation systems; trained model can only learn about areas of the features space that have been explored by previous models. This type of uncertainty, in contrast to data uncertainty, can be reduced if exploration is directed into areas of the feature space that were unexplored, making future models more robust to diverse types of recommendations. We estimate model uncertainty using the Monte Carlo dropout method as a bayesian approximation introduced at [6]. Specifically, we train our DNN model using dropout and during inference we perform T stochastic forward passes through the network, where T is a tunable parameter. We collect T estimations $\hat{y}_i$ and estimate model uncertainty as follows:

$$\sigma = \sqrt{\frac{1}{T} \sum_{i=1}^{T} \hat{y}_i^2 - (\frac{1}{T} \sum_{i=1}^{T} \hat{y}_i)^2} \qquad (4)$$

4.4 Optimistic strategy

Simple algorithms like ϵ-greedy choose actions indiscriminately during exploration, with no specific preference for targets that have higher probability to be successful in exploitation. Uncertainty estimations allow to extend ϵ-greedy and employ the upper confidence bound (UCB) algorithm for better and adaptive exploration of new targets. For example, UCB will prioritize targets with titles that are composed of words that weren't previously recommended (via model uncertainty) and targets that have a larger variability in the CTR of their features (via data uncertainty).

Our marketplace is defined by a very high recommendation turnover rate, with new content being uploaded every day and old one becoming obsolete. We allocate ϵ percent of our recommendation traffic to UCB; We estimate both the mean payoff μ^t and the standard deviation σ^t of each target t and select the target that achieves the highest score A where a is a tunable parameter.

$$A = \arg \max_t (\mu^t + a \cdot \sigma^t) \qquad (5)$$

5 EVALUATION

This section contains two sets of results. First, we evaluate the effect of DDN modeling and of the various types of uncertainties, showing intuitive examples. Next, we show the impact of integrating DDN into Taboola's online recommendation engine.

5.1 Uncertainty estimations

For the results that follow in this subsection we have trained our models employing only title as feature vector for the target making the results human interpretable. In Fig. 4 we show the mean data uncertainty after bucketizing targets according to the number of times r they have been recommended. The data uncertainty of the MDN model depends on r, i.e. low r leads to high data uncertainty. This is an undesirable correlation; previously trained models chose to show more times recommendations from specific areas of the feature space, leading to reduced measurement uncertainty in the training set for those examples. MDN doesn't account for measurement noise explicitly, which causes a pollution of data uncertainty estimates. In contrast, DDN accounts for measurement uncertainty
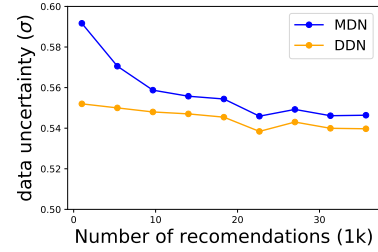

Figure 4: Comparison between MDN and DDN models showing the effect of explicitly modeling measurement noise on data uncertainty.
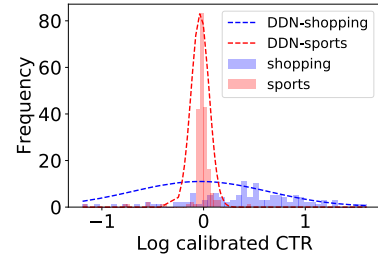

Figure 5: Log calibrated CTR distribution for two targets groups (shopping and sports) with DDN's data uncertainty estimation for a random target from each group.

explicitly and is able both to reduce the predicted uncertainty and the aforementioned correlation significantly. This highlights the benefit of decorrelating measurement noise from data uncertainty (see sec. 4.2).

In Fig. 5 and Table 1 we study the nature of data uncertainty in the context of recommender systems. We first selected two groups of targets related to shopping and sports where intra-group targets are semantically close (see Table 1). Further, we depict the CTR histogram of the two groups together with the distribution induced by the DDN prediction for one randomly selected target from each group (Fig. 5). We observe that the shopping group has large variability in CTR, due to the fact that the specific product that is being advertised highly affects CTR. In contrast, the sport group have relatively consistent CTR. We observe that the DDN model is able to capture and model this target-specific variability and thus have the ability to exploit it in the optimistic strategy.

As discussed in sec. 4.3, model uncertainty should capture what the model doesn't know. In order to validate this, we perform Kernel Density Estimation (KDE) over the targets' title feature representation in the training set, enabling us to quantify the semantic distance for each target from the training set. This way, targets located far away from the training examples i.e. belong to areas of the feature space which were less explored will have low Probability Distribution Function (PDF) value. In Fig. 6 we depict model uncertainty estimations for the DDN model after bucketizing targets in the validation set according to their PDF value relative to the training set. We observe that model uncertainty, is anti-correlated to the PDF value of the targets, indicating that DDN model indeed estimates

shopping	sports
TOMS For $35 - 41% Off	Benfica vs Manchester United Betting
Nike For $72 - 40% Off	It's The Only Way To Watch The Premier League
Magnaflow Performance Mufflers From $68.91	LIVE: Arsenal vs Norwich City
Sun Dolphin Mackinaw 15.6' Square Back Canoe	Premier League Castoffs Starting Over at Age 11
Brooks For $65 - 35% Off	Real Madrid Held to Draw With Tottenham
ASICS For $50 - 29% Off	Rush for a 32/32 Score. NFL Team + City Match

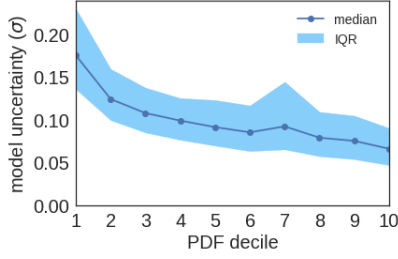**Table 1: Indicative targets for the two selected groups (shopping and sports)**

Figure 6: DDN model uncertainty estimation after bucketizing targets according to their PDF values calculated using KDE over the training set. Low PDF value corresponds to big semantic distance from training set i.e. unexplored areas of the feature space.
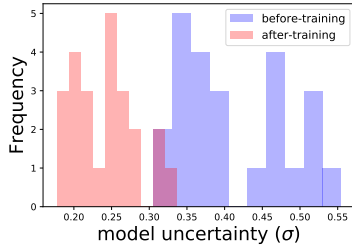

Figure 7: Distribution of DDN model uncertainty estimation on selected group of targets, before and after employing during training the target "BMW X5".

high model uncertainty in less explored areas of the features space, which is a desirable behaviour in recommender systems.

Another interesting observation (Fig. 7 and Table 2), is how model uncertainty is being affected by adding to the training set targets from unexplored areas of the feature space. We selected a group of targets related to car advertisement with low PDF values and high model uncertainty. We added one target from the group ("BMW X5") to the training set and retrained the model. We observe a reduction in the estimated model uncertainty, indicating that proactively exploring targets with high model uncertainty can indeed lead to model uncertainty reduction.

5.2 Performance evaluation

Data: We use the browsed website (i.e. *publisher*) as the user context for the following experiments. In all of the experiments we

car related targets
2011 BMW M3
2005 Jaguar XK-Series XK8 Roadster
2017 BMW X5
Mazda MX-5 Miata
BMW X6
Find a BMW X5 Near You!
The Fastest Car BMW i8

Table 2: Indicative targets for the selected group.

used three months of historical data for training, containing ~10M records of target-publisher pairs. The dataset contains ~1M unique targets and ~10K unique publishers. Every offline experiment has been run on multiple time slots to validate that the results were statistically significant.

Models: In all models we performed an extension of the $\epsilon-greedy$ algorithm, where we allocate ϵ percent of the recommendation traffic to targets that have not been heavily exploited previously by the recommendation algorithm.

1. **REG** corresponds to our deep model described in sec. 3, where the output is just the predicted CTR scalar employing MSE as loss as opposed to a GMM.

2. **MDN** is similar to REG, with the GMM layer and the use of the optimistic strategy introduced in sec. 4.4.

3. **DDN** is similar to MDN, with the measurement uncertainty modeling and optimistic strategy secs. 4.2, 4.4 .

In order to have a fair comparison, we tuned the hyper-parameters (e.g. embedding sizes, number of layers, number of mixtures) for each model separately; we performed thousands of iterations of random search, and chose the parameters that yielded the best results. We have found that this hyper-parameter tuning procedure was crucial in order to get the best possible results from our models, both offline and online.

Metrics and evaluation: we use Mean Squared Error (MSE) for offline evaluation of our models. Due to the dynamic nature of online recommendations it is essential that we evaluate our models online within an A/B testing framework, by measuring the average RPM. In addition, we utilize an online throughput metric which aims to capture the effectiveness of the exploration module; this metric counts the number of new targets that were discovered by the exploration mechanism at a certain given day by being shown significantly for the first time. We expect that exploration models which are better at exploring the feature space will learn to recommend more from this pool of new targets. Similarly, we have

Model	REG	MDN	DDN
RPM lift	0%	1.2%	2.9%

Table 3: A comparison of the online RPM lift between the different models

Dataset	MDN	DDN	Improvement
D1	0.2681	0.25368	5.3%
D2	0.25046	0.24367	2.7 %

Table 4: MSE CTR estimation and relative improvement of DDN over MDN in the MSE when trained over two datasets that differ by the amount of measurement noise.

a	0	0.5	1	1.5
RPM lift	0%	-0.05%	-0.2%	-0.3%
Target throughput lift	0%	6.5%	9.1%	11.7%
Advertiser throughput lift	0%	2.1%	3.7%	5.1%

Table 5: RPM lift, targets and advertisers throughput as a function of different values of a.

a metric for advertisers throughput. In addition to RPM dynamics, maintaining high throughput levels is essential to ensure advertiser satisfaction levels.

5.2.1 Experimental results. **Model comparison:** in Table 3 we compare the three aforementioned models in terms of online RPM. We observe that both **MDN** and **DDN** outperform **REG** by 1.2% and 2.9% respectively. Although the improvements may seem small numerically, they have a large product impact as they translate to significantly higher revenue. In addition, it's noteworthy that **REG** is a highly optimized and tuned model which is our current state-of-the-art model making it a very competitive baseline to win. These results verify once again that the loss attenuation achieved during training has enabled the model to converge to better parameters, generalizing better to unseen examples. Furthermore we observe that **DDN** outperforms **MDN** by 1.7%, indicating that deconvolving measurement noise from the data uncertainty leads to further gains.

Measurement noise: in Table 4 we compare the MDN and DDN models by training them on D1 and D2 datasets. D1 contains more noisy data points with relatively small amount of empirical data, while D2 contains examples with higher empirical statistical significance. We observe that DDN improves on MDN performance by 2.7% when using D1 for training, and by 5.3% when using D2. This validates that integrating measurement noise into our modeling is crucial when the training data contains very noisy samples, by attenuating the impact of measurement noise on the loss function.

RPM lift vs. targets throughput: we analyzed the effect of the parameter a found in eq. 5 by employing data uncertainty. From a theoretical standpoint, increasing this value is supposed to prioritize higher information gain at the expense of RPM, by choosing targets with higher uncertainty. This trade-off is worthwhile in the long term. In Table 5 we observe that there is an inverse correlation between RPM and throughput which is triggered by different values

of a, with targets and advertisers throughput increasing by 11.7% and 5.1% respectively when setting $a = 1.5$. Choosing the right trade-off depends on the application and the business KPIs. For our case we chose $a = 0.5$, resulting in a good throughput gain with a small RPM cost.

6 CONCLUSIONS

We have introduced Deep Density Network (DDN), a hybrid unified DNN model that estimates uncertainty. DDN is able to model non-linearities and capture complex target-context relations, incorporating higher level representations of data sources such as contextual and textual input. We presented the various types of uncertainties that might arise in recommendation systems, and investigated the effect of integrating them into the recommendation model. We have shown the added value of using DNN in a multi-arm bandit setting, yielding an adaptive selection strategy that balances *exploitation* and *exploration* and maximizes the long term reward. We presented results validating DDN's improved noise handling capabilities, leading to 5.3% improvement on a noisy dataset. Furthermore, DDN outperformed both REG and MDN models in online experiments, leading to RPM improvements of 2.9% and 1.7% respectively. Finally, by employing DDN's uncertainty estimation and optimistic strategy, we improved our exploration strategy, depicting 6.5% and 2.1% increase of targets and advertisers throughput respectively with only 0.05% RPM decrease.

REFERENCES

[1] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47, 2-3 (2002), 235–256.
[2] Christopher M Bishop. 1994. Mixture density networks. *Technical report* (1994).
[3] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. 2015. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424* (2015).
[4] G. Chamiel, L. Golan, Rubin A., M. Sinai, A. Salomon, and A. Pilberg. 2013. Click through rate estimation in varying display situations. *United States Patent Application Publication* (2013).
[5] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.
[6] Y. Gal and Z. Ghahramani. 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Intl' Conf. on machine learning*. 1050–1059.
[7] Jacob Goldberger and Ehud Ben-Reuven. 2017. Training deep neural-networks using a noise adaptation layer. In *ICLR*.
[8] A. Kendall and Y. Gal. 2017. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? *arXiv preprint arXiv:1703.04977* (2017).
[9] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
[10] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*. ACM, 661–670.
[11] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7, 1 (2003), 76–80.
[12] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.
[13] V. Mnih and G. Hinton. 2012. Learning to label aerial images from noisy data. In *Proc. of the 29th Intl Conf. on Machine Learning (ICML-12)*. 567–574.
[14] Radford M Neal. 2012. *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media.
[15] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082* (2014).
[16] W. R Thompson. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* 25, 3/4 (1933).
[17] L. Zhu and N. Laptev. 2017. Deep and Confident Prediction for Time Series at Uber. In *Data Mining Wrkshp*. IEEE, 103–110.