Probabilistic Approach for Recommendation Systems

1st Nada Abdalla
Amazon Web Services

Seattle, Washington, USA
nadabdal@amazon.com

2nd Damien Forthomme
Amazon Web Services

Seattle, Washington, USA

***Abstract*—In this article, we propose a new probabilistic approach for product recommendations using deep learning framework, combining information from historical observations, similar users and prior knowledge. The deep learning approach is using autoregressive recurrent networks to model the recommendations probabilistically from a Bernoulli distribution. If prior information exists we implement a Pseudo-Bayesian approach, where we obtain posterior samples assuming Bernoulli likelihood on the sampled data from the deep learning model. The proposed approach allows for a very flexible modeling of product recommendations and quantifying uncertainty in predictions. Simulations and experiments were conducted to demonstrate the applicability and performance of the model. Comparisons made to related recommendation models revealed more accurate predictions among the proposed models.**

***Index Terms*—DeepAR; Monte Carlo; LSTM; Recommender Systems; EM**

I. Introduction

We consider the problem of developing a probabilistic, time aware item recommender system. In this setting, the broad scientific goal is to capture the underlying relationships generating user choices by combining user history if available and interactions, in addition to expert knowledge if available.

Many traditional approaches to recommendations use item-to-item similarity, assuming that items that are clicked/purchased together are correlated. While highly interpretable and effective in many settings [1], these models ignore uncertainty, heterogeneity between the users and the effect of time on the user behavior. Other methods look at user-to-user similarity and take into account historical user behavior [2]. Both approaches lack the ability to produce time aware recommendations and the ability to work on cold start problems. They also lack the ability to quantify uncertainty in a probabilistic manner. As far as we know, there has not been any work prior to this work on probabilistic deep learning models in the context of recommendation systems. Recent work on probabilistic modeling mostly focused on forecasting problems. For example, [3] assumed Gaussian or Negative Binomial likelihoods to predict the probability distribution of the forecasted estimates in what they refer to as DeepAR. On the other hand, deep learning based recommendation models have focused on enhancing the estimated recommendations by modeling the non-linear relationships between the features and not on probabilistic inference. The approach proposed by [4] uses parallel RNNs to model different features, where hidden states of these networks are merged to produce the score for all items. Another form of RNN recommendation models includes the use of hierarchical RNN (HRNN) to evolve the latent states across the users' history [5]. In addition, some recent work considered enhancements to RNNs which aims at avoiding noisy signals and overfitting. For instance, [6] used data augmentation approach and embeddings dropouts. These modifications however may only be beneficial for session based recommendations. Another interesting recommendation model was introduced by [7] where recommendation is posed as extreme multiclass classification and the prediction problem becomes to accurately classify a specific video watch among millions of videos based on a user and context. The deep neural network tries to learn the user embeddings as a function of the user's history and context that are useful for discriminating among videos with a softmax classifier. More recent advancements in sequential recommendation models consider bidirectional architecture to model user behavior sequences, such as BERT4Rec [8].

Bayesian deep learning methods are highly flexible, but more complex. In this context, Bayesian methods [9] have an established history of success in combining prior information and direct measurements to quantify uncertainties through posterior inference. Bayesian probabilistic estimation can help with a lot of scenarios that require expert knowledge such as cold starts. User and/or item cold start is a common problem that is usually addressed dependent on the user-item preferences. [10] proposed using item based stereotypes, in a pre-processing step into the recommendation model.

This approach however requires separate modeling for stereotype creation, while in the Bayesian framework it is built into the prior specifications. The Bayesian approach also allows for incorporating expert knowledge on an item, which is very common in industries such as medical device personalization, software services and others. [11] has proposed using expert inputs in physicians surveys to identify patient characteristics relevant to the performance of the medical device tasks. The proposed approach suffers from lack of generalization to other applications and quantification of uncertainty.

The importance of recommendation models lies in their ability to produce estimates of the usage likelihood using information generated by the observations. We extend upon the autoregressive (deep) neural network model proposed by [3], where we model the likelihood function for the estimated outcomes using Bernoulli distributions. We can enrich the inference on the predicted probability using a Beta prior which enables incorporating expert/prior knowledge if available in a Bayesian like framework. We refer to our method as Pseudo-Bayesian, where rather than fitting a complete Bayesian Neural Network that can be computationally intensive [12], we apply the Bayesian framework on the sampled/simulated Bernoulli data resulting from the deep neural network. Approximate Bayesian methods such as Approximate Bayesian computation (ABC) [13] are rapidly gaining popularity, nonetheless they make assumptions and approximations whose impact needs to be evaluated. The proposed model will yield a more flexible and robust recommender system that outputs a probability distribution rather than a single point estimate. The uncertainty in the prediction can be directly quantified through the probability distribution, which is very important especially in item ranking problems. We discuss a general framework for inference and demonstrate our approach on simulated data and real industry data. The focus of the paper is on one step recommendation, however the methodology can easily be extended to multiple steps in the future. [1]

II. Related Work

A. DeepAR

[3] proposed DeepAR, a methodology that makes probabilistic forecasts in the form of Monte Carlo samples that can be used to compute consistent quantile estimates for all sub-ranges in the prediction horizon. It is based on training an autoregressive recurrent neural network model on a large number of related time series. Using data from related time series allows more complex models to be fitted without over-fitting and provides forecasts for items that have little or no history available. It does not assume Gaussian noise, but can incorporate a wide range of likelihood functions, allowing the user to choose one that is appropriate for the statistical properties of the data.

We approach the recommendation problem by incorporating an appropriate likelihood such as the Bernoulli likelihood, and combining that with non-linear data transformation techniques, as learned by the (deep) neural network.

Suppose that $\{z_t : t \in \mathcal{T}\}$, $z_t \in \Re^{n_z}$ is a stochastic process, where $\mathcal{T}$ is the index set and n_z is the dimension of the target vector. Let DNN_t represents the neural network used in the model at time t. At each time t, the inputs to the network are the covariates X_t, the target at the previous time step z_{t-1} and the hidden state from the previous network output layer h_{t-1}. The model output layer at time t thus would be $h_t = DNN_t(h_{t-1}, z_{t-1}, x_t)$, where x_t is the feature set at time t, h_t is then further used to calculate parameters of the likelihood such as, μ and σ in case of Gaussian distribution or θ in case of Bernoulli distribution, which is used for training the DNN parameters. The DNN initial cell state and hidden state are initialized with zeroes. In general settings, we would want to find Ω that maximizes the log likelihood function l, where $l(z|\Omega) = logL(z|\Omega)$, where Ω is the set of the DNN parameters and the likelihood parameters.

Our goal is to model the conditional likelihood at time $t = T + 1, \dots T^*$, using the past at times $t = 0, \dots T$, assuming the covariates are known for all time points. Figure (1) is a representation of the training and prediction architectures.

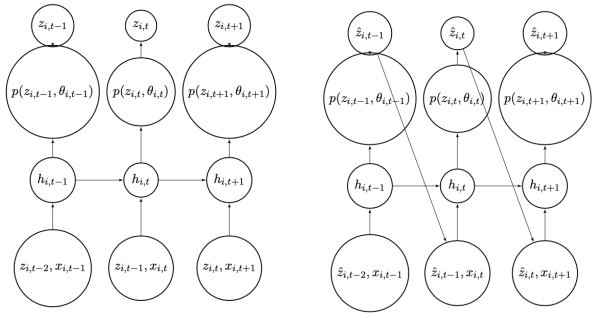

Fig. 1: Graphical representation of DeepAR training and prediction

In prediction, we can obtain samples directly from the target distribution which is a Bernoulli distribution in the recommendation framework.

[1] We will publish our code as a Sagemaker prebuilt Docker image to promote reproducibility and expand the impact of our research.

B. Recommendations as Multiclass Classifications

[7] used deep neural networks for Youtube recommendations. They posed recommendation as extreme multiclass classification where the prediction problem becomes to accurately classify a specific item w_t at time t among millions of items i from a corpus V based on a user U and context C. Using the above definitions, the probability of item i at time t becomes

$$P(w_t = i|U, C) = \frac{e^{v_i u}}{\sum_{j \in V} e^{v_j u}}$$

where $u \in R^N$ is a high-dimensional embedding of the user and context pair and the $v_j \in R^N$ represents embeddings of each candidate item. The embeddings are learned jointly with all other model parameters through normal gradient descent back propagation updates. Features are concatenated into a wide first layer, followed by several layers of fully connected Rectified Linear Units (ReLU).

III. Model

In this section we describe the proposed Pseudo-Bayesian DeepAR model for recommendations. In Section III-A we introduce DeepAR for recommendations, then we explain the Bayesian addition to the model in Section III-B.

A. DeepAR for recommendations

Our model is based on multiclass classification presented in Section II-B, where the output is a probability distribution as outlined in Section II-A. In recommendation systems, we can use the Bernoulli likelihood, since for every user i at time t, we would like to suggest whether an item should be recommended or not. The likelihood is defined as

$$L(Z|\theta) = \prod_{i,t} \theta_{i,t}^{z_{i,t}} (1 - \theta_{i,t})^{1-z_{i,t}}, z_{i,t} = 0, 1,$$

$$\theta_{i,t} = \theta(h_{i,t}) = 1/(1 + exp - (\omega_\theta^T h_{i,t} + b_\theta)),$$

where the parameter θ is obtained from the DNN output layer $h_t = DNN_t(h_{t-1}, z_{t-1}, x_t)$, where h_t is the output layer at time $t-1$, and transformed through an affine function followed by softmax function to ensure that it lies between 0 and 1, and ω and b are parameters of the affine function of the network output. DeepAR for recommendations is provided in Algorithm 1. The algorithm emulates Expectation Maximization (EM) algorithm [14] since we are optimizing a likelihood function with the objective of learning a probability distribution (Bernoulli in this case).

Formally, assume we have $i = 1, \ldots, N$ users, $t = 0, \ldots T$ time points and $j = 1, \ldots M$ items to recommend. We define the dependent variable $z \in \{0, 1\}$ as

$$z_{i,j,t} = \left[\begin{array}{c} 1 \text{ if user i uses item j at time t} \\ 0 \text{ otherwise} \end{array} \right]$$

such that Z is the $(MN) \times T$ matrix.

Similarly, we define the model covariates X as the $NM \times T^* \times R$ array, where R is the number of features and $T^* \geq T$. For large data sets, we can use embeddings for some or all the variables [15], which is believed to improve model performance and reduce the dimension of the feature matrix.

For a given time t, we obtain $s = 1, \ldots S$ Monte Carlo samples from the target distribution, i.e $\hat{z}_{i,j,t,s}$. Using the above S samples, we can calculate

$$\hat{\theta}_{i,j,t} = \sum_{s=1}^{S} \hat{z}_{i,j,t,s}/S \tag{1}$$

In a frequentist settings, we can use the Central Limit Theorem, by using the Normal approximation to the binomial to get the upper and lower bounds of the predicted probability.

$$UL_{0.95} = \hat{\theta} + 1.96 \times \sqrt{\hat{\theta}(1 - \hat{\theta})/S},$$
$$LL_{0.05} = \hat{\theta} - 1.96 \times \sqrt{\hat{\theta}(1 - \hat{\theta})/S} \tag{2}$$

In the Pseudo-Bayesian framework, as explained in the following section, we will generate the credible intervals using posterior samples for the probability θ. The estimated probabilities can be used to rank the recommendations for the given user, item and time.

Algorithm 1 DeepAR for Recommendations

- Given $x_0^i, \ldots, x_T*^i$, for $i = 1, \ldots, N$,
  - For $(t = 0)$, we initialize $h_{i,0} = 0$ and $z_{i,0} = 0$ such that $h_{i,1} = DNN(h_{i,0}, z_{i,0}, x_{i,1}, \Omega)$ where Ω is the DNN parameter vector and the likelihood parameter θ.
  - For $t = 1, \ldots, T$, the target distribution is $z_{i,t} \sim p(.|\theta(h_{i,t}, \Omega))$, where $h_{i,t} = DNN(h_{i,t-1}, z_{i,t-1}, x_{i,1}, \Omega)$.
  - And for $t = T + 1, \ldots T^*$ the target distribution is $\hat{z}_{i,t} \sim p(.|\theta(\hat{h}_{i,t}, \Omega))$ where $\hat{h}_{i,t} = DNN(h_{i,t-1}, \hat{z}_{i,t-1}, x_{i,1}, \Omega)$ initialized with $\hat{h}_{i,t} = h_{i,t-1}$ and $\hat{z}_{i,t} = z_{i,t-1}$, where $t = T + 1$.
- At each iteration the parameters of both the DNN and the likelihood, are learned by maximizing the log likelihood $l = \sum_{i=1}^{N} \sum_{t=0}^{T} z_{i,t} log\theta_{i,t} + (1 - z_{i,t}) log(1 - \theta_{i,t}))$ using stochastic gradient descent with respect to all the prameters Ω.
- Draw S samples from the target distribution.
- Develop probabilistic inference from the sample draws.

B. Pseudo-Bayesian methods

We extend upon the model in section III-A by using the Bernoulli sampled data and imposing a prior distribution on the predicted probability and drawing inference using the posterior distribution. The use of a Bayesian framework allows very flexible modeling of the predicted probability for a particular item. We use conjugate prior on the probability due to their algebraic convenience as well as intuitively showing the updates on the prior.

For the Bernoulli likelihood, the Beta distribution is a conjugate prior, where the posterior distribution is also a Beta distribution. More generally, the proposed model admits the following hierarchical representation

$$\begin{aligned} z_{i,j,t,s}|\theta &\sim Bernoulli(\theta), \\ Z_{i,j,t}|\theta &\sim Binomial(S,\theta), \\ \theta &\sim Beta(a,b) \end{aligned} \tag{3}$$

where $Z_{i,j,t} = \sum_s z_{i,j,t,s}$.

Reasonable informative priors can be used where we are able to specify a prior point estimate θ_0 for θ based on previous data i.e. $\theta_0 = z_{old}/n_{old}$, published study, data from a similar experiment, expert knowledge or reasoning about the circumstances. A lot of the recommendation applications can get advantage from expert knowledge on new items and/or users, which can help with cold start problems. Examples of such applications include personalizations in the medical field, where specialists knowledge is available and can help with enhancing the accuracy of the recommendations. Another example is online sales recommendations, where marketing and sales professionals can provide insights on new items.

Given θ_0, the mean of $\theta \sim Beta(a,b)$ is $\theta_0 = \frac{a}{a+b}$, hence, given an estimate of the variance $\sigma^2 = \frac{a}{a+b}\frac{b}{a+b}\frac{1}{a+b+1}$, we can solve for a and b in (3).

$$\begin{aligned} \sigma^2 &= \theta_0(1-\theta_0)\frac{1}{1+a+b}, \\ a+b+1 &= \frac{\theta_0(1-\theta_0)}{\sigma^2}, \\ a+b &= \frac{\theta_0(1-\theta_0)}{\sigma^2} - 1 \end{aligned}$$

such that $\theta_0(1-\theta_0) > \sigma^2$. Finally we get $a = (a+b)\theta_0$ and $b = (a+b)(1-\theta_0)$. Specifying a prior sample size $n_0 = a+b$ and $\theta_0 = a/(a+b)$, we get $a = \theta_0 n_0$ and $b = (1-\theta_0)n_0$. In the absence of informative prior, we can specify $a = b = 1$ which is equivalent to Uniform$[0,1]$ distribution.

C. Model Implementation and Evaluation

We will consider two variations of the model. We will refer to the first as Frequentist Probabilistic Recommendation model (FPR). FPR results from a non-Bayesian framework, where we use (2) to calculate confidence intervals. Our second model imposes a Beta prior and is referred to as Bayesian Probabilistic Recommendation model (BPR). Depending on the application, we can consider a non-informative prior where a and b in (3) equals 1 or informative priors where we specify a and b from previous experiments. We compared the results against several baselines that are grouped into three categories; simple architecture models, deep learning based models and sequential models. The first category includes item-to-item similarities (SIMS) model that uses item based Collaborative Filtering [1] which identifies the co-occurrence of the item in user histories in the interaction dataset to recommend similar items. We also include Matrix Factorization (MF) [16] which represents user/item as a vector of latent features projected into a shared feature space where the user-item interactions are modeled using the inner product of user-item latent vectors. The second category includes Neural Collaborative Filtering [17] which replaces the inner product with a neural architecture that can learn an arbitrary function from data. The third category includes BERT4Rec [8] and hierarchical recurrent neural network (HRNN) [5] which includes the use of RNN to evolve the latent states across the users' history.

For models comparisons, we use Mean Reciprocal Rank (MRR) and precision. MRR tries to measure the rank of the first relevant item. For each user i, a list of recommendations are generated with rank $k_{i,j}$ where $i = 1,\dots N$ and $j = 1,\dots M$. Let the reciprocal of the rank of the first relevant recommendation for user i be $1/k_i$. MRR is the average of the reciprocal rank for all the users, i.e. $MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{k_i}$. MRR can be generalized to the relevance of the top L items, i.e $MRR = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{L}\frac{relevance_j}{k_{i,j}}$. We consider MRR at 25 as well as precision at 25 which means the fraction of relevant items in the top 25 recommendations.

IV. Data Analysis

In this section, we evaluate the performance of the models discussed in Section III-C, using computer-simulated datasets as well as two real industry data sets. The first data set contains users consuming streaming content on Twitch and we will refer to it as Twitch data [18]. The second data set consists of AWS users used products and we will refer to it as AWS data set. The effectiveness of the methods proposed are assessed through the measures discussed in Section III-C. Moreover, we compare the performance of the proposed models to the SIMS, MF, NCF, BERT4Rec and HRNN models. We consider prior settings for the BPR model based on results of previous experiments if exists as explained in the following section. The models are

implemented using Torch using ml.g4dn.8xlarge AWS EC2 instance with 32 CPUs and up to 4 GPUs. We optimize the neural models for the likelihood loss using Adam optimizer [19]. For each data set, we used manual search to select LSTM hyperparameter candidates (e.g hidden units, dropout rate). We fit the model on the data set up to the last time point before the forecast interval, and picked the candidate with highest evaluation metric. In all cases hidden units were between 5 and 10, dropout regularization [20] was between 0-0.2 and best learning rate in all applications was found to be 0.001.

A. Prior Settings

In recommendation systems, reasonable informative priors may exist in some applications based on expert knowledge or prior experiments. For the simulation data set and Twitch data set, we assume a neutral prior in the form of $Beta(1/3, 1/3)$, which leads to posterior distributions with approximately 50 percent probability that the true value is either smaller or larger than the maximum likelihood estimate as proved in [21]. For AWS data set we use historical industry specific data to construct informative priors as follows. The posterior Beta mean is convex weighted mean of the prior mean and data mean, where the weights are $w = (a+b)/(a+b+N)$ and $N/(N+a+b)$ respectively, where N is the total number of recommendations samples for a given user and item. So it depends on the relative information in the prior $(a + b)$ and the data N. We assume that $a + b = N/4$, which puts more weight on the data than on the prior. Also we assume that if an item was observed in a historical experiment z times out of N^* times, the prior mean is $a/(a + b) = z/N^*$, hence $a = z/N^* \times N/4$. If the item was not observed, we assume the non-informative prior $Beta(1, 1)$.

B. Data

1) Simulation Data: We generated a data set of M=50 items, and N=1000 users over T=24 time points by assuming a correlation structure between the users and items. We generated random variables $\{y_{i,j}, i = 1, \dots N, j = 1, \dots M\}$ from a multivariate Gaussian distribution with 0 mean and covariance Σ_{MN}, where $\Sigma_{ij,i*j*}$ is the correlation between user i, item j and user i^*, item j^* at time t. The correlation structure is random and is not pre-specified. We then generate $\{z_{i,j} \sim Bernoulli(p = e^{y_{i,j}}/(1 + e^{y_{i,j}})), i = 1, \dots N, j = 1, \dots M\}$ as the outcome of interest. To generate the sequence of observations over time $t = 1, \dots T$, we assume a Markov model with a 2×2 transition probability matrix P, where the transition states are 0 and 1. We assume that $P = \begin{bmatrix} 0.8 & 0.2 \\ 0.05 & 0.95 \end{bmatrix}$. The transition probability matrix assumes that the likelihood that the user is adopting a new item is 20% and that if the user uses a particular item, they are 95% likely to keep using it.

2) Real Data: The first data set contains users consuming streaming content on Twitch. This is a dataset of 100k users consuming streaming content on Twitch. We retrieved all streamers, and all users every 10 minutes during 43 days. We eliminated items with interactions ≤ 3 and users with ≤ 15 sessions. For a given set of users in the given time period with all 0 interactions, we randomly assigned interactions using the most popular streamers. This helps removing the noise by keeping users and/or items with sufficient information for modeling. This makes the data more session-based where recommendations are based mostly on the interactions in the current user session rather than on the user's history. The features included in the model are streaming time and the user identifier. The second data set consists of 30k AWS customers and the products they use on AWS over a period of 24 months, along with some metadata on the customers.

C. Results

Table I presents the precision and MRR for the test set. We calculate metrics by comparing the recommendations the solution version generates to the actual interactions in the newest 10% of each user's data from the testing set. The results in Table I shows that, in absence of informative priors, BERT4Rec model's performance is comparable to probabilistic based models. When we use informative priors such in AWS data, BPR outperforms all the models. SIMS model is a very competitive outperforming the more sophisticated models in precision followed NCF.

V. Discussion

We have proposed a probabilistic framework for recommendation systems. This approach provides a very flexible yet robust modeling of recommendation systems such that any user based recommendation data can be accommodated. The probabilistic framework provides a natural approach for uncertainty quantification in various ways. The Bayesian framework proposed allows imposing prior distribution based on expert knowledge or previous experiments. In our applications, sequential methods tended to perform better since user identifiers are present and propagate information from the previous user session to the next, thus improving the recommendation accuracy. BPR and FPR tended to perform very similarly when we use non-informative prior with respect to precision and MRR with some improvement when we use informative prior from historical experiments.

The model has some limitations in providing predictions for more than one time point in the future since it uses previous time point estimate which can

TABLE I: MRR@25 and Precision@25 for the SIMS, MF, NCF, HRNN, BERT4Rec, FPR and BPR models for AWS, Twitch and Simulation Data sets

	AWS Data	
Model	MRR@25	Precision@25
SIMS	1	0.80
MF	0.43	0.36
NCF	1	0.74
HRNN	0.7	0.43
BERT4Rec	1	0.78
FPR	1	0.86
BPR	1	0.90
	Twitch Data	
Model	MRR@25	Precision@25
SIMS	1	0.80
MF	0.50	0.46
NCF	1	0.48
HRNN	1	0.33
BERT4Rec	1	0.99
FPR	1	0.99
BPR	1	0.99
	Simulation Data	
Model	MRR@25	Precision@25
SIMS	0.85	0.28
MF	0.57	0.53
NCF	0.95	0.55
HRNN	0.94	0.30
BERT4Rec	1	0.80
FPR	1	0.80
BPR	1	0.80

propagate the errors over time. We conclude with some indicators for future research. First, as alluded to earlier, we can include training the model on the predicted values from previous time points rather than on the actual observations, such that the model would generalize better for predicting different time points in the future. We can also enrich the Bayesian inference by fitting a complete Bayesian neural network, incorporating more informative priors from experts and/or relax the assumption for conjugate prior which can provide richer Bayesian framework overall. We can consider Quasi-Bayesian [22], an alternative decision theory framework that deals with difficulty of choosing a single prior when we might actually have many preferences/distributions over thetas with a complex or unknown relationship to the state of the world and previous "decisions"/realizations.

REFERENCES

[1] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[2] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan, *Collaborative filtering recommender systems*. Now Publishers Inc, 2011.

[3] J. G. David Salinas, Valentin Flunkert and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, pp. 1181–1191, 2020.

[4] B. Hidasi, M. Quadrana, A. Karatzoglou, and D. Tikk, "Parallel recurrent neural network architectures for feature-rich session-based recommendations," in *Proceedings of the 10th ACM conference on recommender systems*, 2016, pp. 241–248.

[5] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, "Personalizing session-based recommendations with hierarchical recurrent neural networks," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, 2017, pp. 130–137.

[6] Y. K. Tan, X. Xu, and Y. Liu, "Improved recurrent neural networks for session-based recommendations," in *Proceedings of the 1st workshop on deep learning for recommender systems*, 2016, pp. 17–22.

[7] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM conference on recommender systems*, 2016, pp. 191–198.

[8] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proceedings of the 28th ACM international conference on information and knowledge management*, 2019, pp. 1441–1450.

[9] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian Data Analysis*. Chapman and Hall/CRC, 2013.

[10] N. AlRossais, D. Kudenko, and T. Yuan, "Improving cold-start recommendations using item-based stereotypes," *User Modeling and User-Adapted Interaction*, pp. 867–905, 2021.

[11] B. M. Knisely, M. Vaughn-Cooke, L.-A. Wagner, and J. C. Fink, "Device personalization for heterogeneous populations: leveraging physician expertise and national population data to identify medical device patient user groups," *User Modeling and User-Adapted Interaction*, vol. 31, pp. 979–1025, 2021.

[12] C. Bishop, *Bayesian methods for neural networks*. United Kingdom: Oxford University Press, 1995.

[13] K. Csilléry, M. G. Blum, O. E. Gaggiotti, and O. François, "Approximate bayesian computation (abc) in practice," *Trends in ecology & evolution*, vol. 25, no. 7, pp. 410–418, 2010.

[14] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977. [Online]. Available: http://www.jstor.org/stable/2984875

[15] C. Guo and F. Berkhahn, "Entity embeddings of categorical variables," *arXiv preprint arXiv:1604.06737*, 2016.

[16] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[17] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering." International World Wide Web Conferences Steering Committee, 2017, pp. 173 – 182.

[18] J. M. Jérémie Rappaz and K. Aberer, "Recommendation on live-streaming platforms: Dynamic availability and repeat consumption," in *RecSys*, 2021.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[21] J. Kerman, "Neutral noninformative and informative conjugate beta and gamma prior distributions," *Electronic Journal of Statistics*, vol. 5, no. none, pp. 1450 – 1470, 2011. [Online]. Available: https://doi.org/10.1214/11-EJS648

[22] F. J. Girón and S. Ríos, "Quasi-bayesian behaviour: A more realistic approach to decision making?" *Trabajos de Estadistica Y de Investigacion Operativa*, vol. 31, no. 1, pp. 17–38, 1980.