Online Learning for Distributed and Personal Recommendations—a Fair approach

Martin Tégner [1 2]

Abstract

We present an interpretable Markov-chain mixture model for recommendations. Each component Markov-chain models how items are related and viewed according to different types of activities. The mixture distribution represents a user-specific blend of such activities. For a personalised model, we use Bayesian statistics and continuously update the user's posterior from a stream of item-views. We propose an online learning algorithm for this purpose. In addition, for fairness and privacy, our approach does not use, collect or store any private information or personal data.

1. Introduction

Making a *recommendation* is the task of choosing a particular item (from a given set of items) to present to a user. The goal is to make suggestions that are relevant: A recommendation ultimately leads to a product purchase, a movie being watched, an app downloaded, or more generally, to some desired action by the user.

Machine learning and data-driven methods have been successfully used for recommender systems. These algorithms are often designed to make recommendations based on similarities in preferences: If item A and B are frequently preferred together among users, a 'new' user will be recommended B as a result of A. This is *collaborative filtering*, see e.g. Linden et al. (2003). General machine learning methods are also used for *learning* a feature space that predicts such preferences: if two users share some common features, their recommendations tend to be similar; Cheng et al. (2016). Recurrent neural networks have also been applied to directly predict the sequential behaviour of item-views, Hidasi et al. (2015).

[1]IKEA Retail Business, Malmö, Sweden [2]Oxford-Man Institute, University of Oxford, UK. Correspondence to: Martin Tégner <martin.tegner@ingka.ikea.com>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020. Copyright 2020 by the author(s).

These algorithm can, simply speaking, be viewed as supervised learning: an input x is associated to an output y, where x may represent previously seen items, user-specific features, personal data etc., while y is a (list of) item(s) that will be recommended. The mapping from input to output, $x \mapsto y$, is then learned from historical data.

Common to these methods is that their generalisation power rely on large data-sets for learning. The mapping can only make a good prediction y_* for an unseen x_* if it has been trained to (x, y)-examples in the 'closure' of x_*. That is, the training process has explored a sufficiently large and dense input space, so the mapping of unseen inputs is smooth. Otherwise, the learned mapping will typically suffer from bias and high variance (over-fitting). Similarly, for feature based mappings, generalisation to new examples will also require informative features. To represent preferences, this is typically private data such as demographic information, but might also included user-specific attributes and historical data.

In this work, we focus our attention on settings with little access to user-data. This because of availability, or more importantly, due to privacy- and 'data-fairness':

- We refrain from using private information and features from users, and also from collecting and storing data for centralised training.
- We propose a method apt for distributed inference and predictions, which only relies locally on the user's most current item-views.

A data-driven mapping $x \mapsto y$ is 'fully' trained to generalise when it has learned to map every corner of the input space, not only for training data, but also such that it interpolates new inputs reasonable well. One perspective of this is that it has fully learned the *collective* behaviour of users through the information encoded in x, while individual variations are filtered out as noise. While learning general (and meaningful) features is the goal for many supervised learning tasks, it raises a question of *personalization* in the context of recommendations: Two users with equal values of x will receive the same recommendation y,[1] even if they might have different intentions with their next item view. To this end, we approach the recommendation task with a probabilistic model designed to dynamically adapt to the user's interactions. In particular, we assume *prior knowledge* about different types of common behaviours, and infer a *personal* mixture over these with Bayesian updating:

- We use a probabilistic mixture model over fixed *item-spaces*, representing different collective behaviours, different intentions or contexts, in the form of Markov chains—a *Markov chain mixture model*.
- We use online learning at the user level to update the mixture from the immediate stream of item-views, to continuously learn the user's individual preferences in real time.

We formalise the recommendation task and present our model along with the online learning algorithm in the next section. We conduct simulation experiments in Section 3 to demonstrate our approach on synthetic data before we conclude in Section 4.

2. Problem formulation and Method

Preliminaries Let $y \in Y$ represent an item with $Y = \{1, ..., M\}$, i.e. we consider a state-space of M possible items that are enumerated $1, ..., M$ (without any meaning attached to order). The goal is a probability distribution over Y which is *individual* to an user: if t is a counter, and $\mathbf{y}_t = (..., y_{t-2}, y_{t-1}, y_t)$ a sequence of items viewed by the user, we will construct a model for

$$p(y_{t+1}|\mathbf{y}_t). \tag{1}$$

This is the conditional distribution over Y *given* historical item views. We like to make recommendations based on (1), for example, by drawing a prediction

$$y_{t+1} \sim p(y_{t+1}|\mathbf{y}_t). \tag{2}$$

To this end, consider a (homogeneous) Markov chain for (1) where $p(y_{t+1}|\mathbf{y}_t) = p(y_{t+1}|y_t)$—this is the Markov property—and

$$p(y_{t+1} = j|y_t = i) = A_{i,j} \tag{3}$$

is the probability of a transition $i \to j$ according to an $M \times M$ transition matrix[2] A.

With a sufficiently long history $\mathbf{y}_t$, we could use maximum likelihood to estimate a user-specific transition matrix (in Bayesian terms, without *prior* knowledge about the user; A is estimated from data $\mathbf{y}_t$ alone). This is, however, not practical: M is typically much larger than the length of $\mathbf{y}_t$, leaving many transition probabilities non-estimated, while (2) would only recommend items that has already been viewed. At the other hand, we could collect the history from all users, and estimate a *common* transition matrix (if we permit such collection of data, that is). This would serve to capture a collective behaviour of users, with individual differences 'averaged out'.[3]

A Markov chain encodes a relationship between all (pairs of) items. Intuitively, it gives a notion of *distance* between all states Y: If the transition $i \to j$ has high (low) probability, the distance from item i to item j is small (large). Similarly, we can think of prior knowledge as relationships between items according to different contexts: If Y represents furniture, a *category-based* context have an armchair closer to a sofa, than a to table-lamp; while a *function-based* context would group items associated with "kitchen" close together in a cluster, with a good distance from a "bathroom" cluster. In this work, we assume a given (collection of) such item-relationships, that are represented as Markov chains. They can be contextual, as illustrated above, inferred from users, or hand-crafted. We think of the Markov chains as representing different types of (fixed) *activities* (i.e. how one item is preferred to another in a stream of item views), and will use Bayesian updating to learn a user-individual *mixture* over these activities.

Model To construct a model both expressive of different activities and adaptive to individual behaviours, consider a given collection of L transition matrices $A^{(1)}, \ldots, A^{(L)}$ where each Markov chain $A^{(k)}$ represents a specific type of activity. At user level, we mix these with a random variable $C \in \{1, ..., L\}$ that represents the user's latent activity, and assume a categorical *prior* distribution $p(C|\pi) = \text{Cat}(C; \pi)$. The parameter $\pi = (\pi_1, ..., \pi_L)$ is a vector of (prior) probabilities over activities:

$$\text{Prob}(C = k) = \pi_k \tag{4}$$

is the user's probability of activity k, i.e. viewing items according to $A^{(k)}$. An activity is thus connected to data

[1]We might re-train our model as soon as we have seen x of the first user, but if this changes the model's recommendations, this hardly adds any *personalization* for the second user.

[2]A:s element are non-negative, and rows sum to one.

[3]Consider two users, the first with n historical transitions $1 \to 2$, the second with n historical transitions $1 \to 3$. An individual A for the first user would assign Prob$(1 \to 2) = 1$, zero otherwise. Similarly, Prob$(1 \to 3) = 1$, zero otherwise, for the second user. The collective A would give an equal transition probability $1/2$ to both state 2 and 3.

through the *likelihood*

$$p(\mathbf{y}_t|C=k) = \prod_{\tau\geq 0} p(y_{t-\tau}|y_{t-\tau-1}, A^{(k)}) = \prod_{\tau\geq 0} A^{(k)}_{y_{t-\tau-1},y_{t-\tau}}$$

which follows from factorisation, the Markov property and (3). Let $\mathcal{L}(\mathbf{y}_t)_k := p(\mathbf{y}_t|C=k)$ denote the likelihood of a particular activity. Bayes' theorem then gives the *posterior*, i.e. the distribution over C given an observed sequence of user-views

$$p(C|\mathbf{y}_t) = \frac{1}{Z} p(\mathbf{y}_t|C)p(C|\pi) \tag{5}$$

with normalising constant $Z = \sum_{k=1}^{L} \mathcal{L}(\mathbf{y}_t)_k \pi_k$. In particular, $C|\mathbf{y}_t$ remains a categorical distribution, with posterior probabilities

$$\text{Prob}(C=k|\mathbf{y}_t) = \frac{1}{Z}\mathcal{L}(\mathbf{y}_t)_k\pi_k =: \pi(t)_k$$

such that $p(C|\mathbf{y}_t) = \text{Cat}(C;\pi(t))$—cf. (4) for the prior.

Online learning Consider next a 'new' sequence of item views $\mathbf{y}_{\text{new}}$ observed at $t' > t$, such that $(\mathbf{y}_t, \mathbf{y}_{\text{new}})$ is the user's full history. Ignoring normalising constants, the new posterior is

$$\begin{aligned} p(C|\mathbf{y}_{\text{new}}, \mathbf{y}_t) &\propto p(\mathbf{y}_{\text{new}}, \mathbf{y}_t|C)p(C|\pi) \\ &= p(\mathbf{y}_{\text{new}}, y_t|C)p(\mathbf{y}_t|C)p(C|\pi) \\ &\propto p(\mathbf{y}_{\text{new}}, y_t|C)p(C|\mathbf{y}_t) \end{aligned} \tag{6}$$

where the second line factorises the likelihood (y_t is the most recent value of $\mathbf{y}_t$; henceforth included in $\mathbf{y}_{\text{new}}$). Notably, in (6) we see that the form of the posterior is preserved, with a new likelihood and 'acting prior' being the *previous* posterior $p(C|\mathbf{y}_t)$. We hence update the categorical posterior distribution with a new probability vector $\pi(t')$ computed from observations $\mathbf{y}_{\text{new}}$ and the previous parameter $\pi(t)$ as

$$\pi(t')_k = \frac{1}{Z}\mathcal{L}(\mathbf{y}_{\text{new}})_k\pi(t)_k$$

where $Z = \sum_{k=1}^{L} \mathcal{L}(\mathbf{y}_{\text{new}})\pi(t)_k$. This makes for an *online* learning algorithm, that updates the posterior continuously with the stream of user data, by using only the user's most recent item-views in each update (as opposed to the entire history of data). We summarise the update step in Algorithm 1. An example of a sequence with probabilities $\pi(t)$ trained by the algorithm is shown in Figure 1 (more details in Section 3).

Algorithm 1 Online posterior update

```
Input: matrices $\{A^{(k)}\}_{k=1}^{L}$, current mixing parameter $\pi$
and data $\mathbf{y}_{\text{new}}$ observed after previous update
for $k = 1$ to $L$ do
  compute $\mathcal{L}(\mathbf{y}_{\text{new}})_k$ from $A^{(k)}$
  update $\pi_k \leftarrow \mathcal{L}(\mathbf{y}_{\text{new}})_k\pi_k$
end for
normalise:
$Z = \sum_k \pi_k$
$\pi_k \leftarrow \pi_k/Z$
Output: updated $\pi$ (last value of $\mathbf{y}_{\text{new}}$, to include in data
of next update)
```

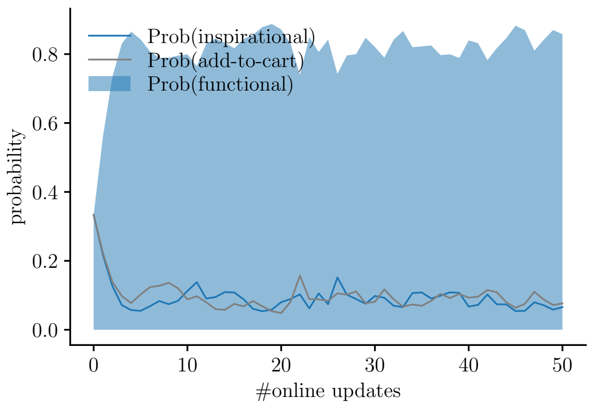

Figure 1. The functional shopper. Posterior probabilities $\pi(t)$ of the categorical mixture distribution over activities, as a function of online updates.

Making recommendations At any point t in time, we make a recommendation based on the user's current posterior $p(C|\mathbf{y}_t)$ (updated continuously with data and parameterized by $\pi(t)$). For this purpose, the posterior predictive distribution for an item $y_{t+1} \in Y$ is given by marginalisation of C

$$\begin{aligned} p(y_{t+1}|\mathbf{y}_t) &= \sum_k p(y_{t+1}, C=k|\mathbf{y}_t) \\ &= \sum_k p(y_{t+1}|y_t, C=k)p(C|\mathbf{y}_t). \end{aligned}$$

Since conditioning on C is a Markov chain, this yields

$$\text{Prob}(y_{t+1}=j|y_t=i,\mathbf{y}_t) = \sum_k A^{(k)}_{i,j}\pi(t)_k. \tag{7}$$

Notice that the data required for computing (7) is y_t, not the entire $\mathbf{y}_t$ (the latter being implicitly encoded in $\pi(t)$) while (7) is a *time-inhomogeneous* transition: the probability of $i \to j$ depends on t through the full history $\mathbf{y}_t$.

With the (categorical!) distribution (7) over Y we may post recommendations by simply drawing a new item and present it to the user.[4] We can also select the distributional mode (the y with highest probability) or using more elaborate strategies based on maximising an expected utility

$$\alpha(y_{t+1}; \mathbf{y}_t) = E_{p(y_{t+1}|\mathbf{y}_t)}[U(y_{t+1}, y_t)]$$

where U is a utility function that assigns a reward to the choice y_{t+1} when the user is currently viewing y_t.

We might also want to recommend a list of items. This can be done by repeated one-step draws from (7), or by drawing a *sequence* from the multi-step predictive posterior

$$\begin{aligned}
&p(y_{t+3}, y_{t+2}, y_{t+1}|\mathbf{y}_t)\\
&\quad= \sum_k p(y_{t+3}, y_{t+2}, y_{t+1}|y_t, C = k)p(C|\mathbf{y}_t)\\
&\quad= \sum_k \prod_{\tau=0}^{2} p(y_{t+1+\tau}|y_{t+\tau}, C = k)p(C|\mathbf{y}_t)
\end{aligned}$$

Again, $p(y_{t+1+\tau}|y_{t+\tau}, C = k)$ is a transition according to $A^{(k)}$. Note that, however, while one-step transitions (7) are Markovian in the sense that the may be expressed as a (t-dependent) transition matrix, the multi-step distribution is not factorising to a 'Markovian' product of one-step transitions. This reveals the mixture model as more general than a single Markov chain.

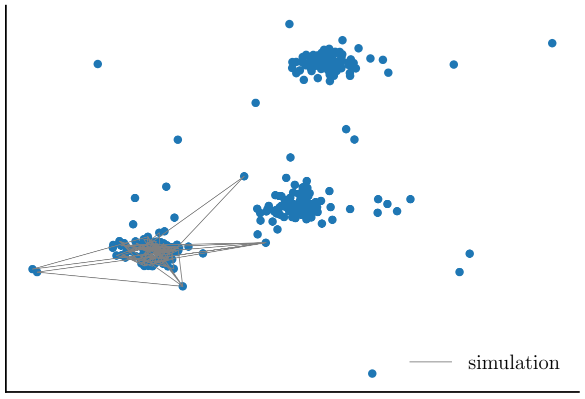

Figure 2. Visualisation (t-SNE) of the functional activity. Each blue dot is a two-dimensional representation of an item based on its transition probabilities in $A^{(1)}$. If two items are plotted close to each other, the transition probability between them is relatively high. Note that there are three clusters with items associated to kitchen, living-room and bathroom, respectively. The grey line shows a sequence $\mathbf{y}$ of 100 simulated items.

[4]In the bandit literature, this is Thompson Sampling.

Posterior tilting A simple modification to the online update of the posterior is *tilting*:

$$\pi(t')_k = \frac{1}{Z}\mathcal{L}(\mathbf{y}_{\text{new}})_k^{\gamma}\, \pi(t)_k^{1-\gamma}$$

where a *forgetting factor* $\gamma \in [0, 1]$ will balance the impact of 'old' data (through $\pi(t)$) and the more current information in $\mathbf{y}_{\text{new}}$. With the case $\gamma = 1$ we can choose to learn only from the user's most recent behaviour, while $\gamma = 0$ would completely ignore the stream of user data.

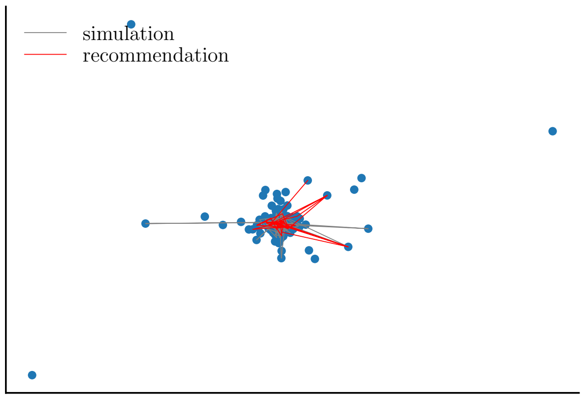

Figure 3. Visualisation (t-SNE) of the user's activity after t_{end}, the final online posterior update. Each blue dot is a two-dimensional representation of an item, constructed from the transition matrix A_{user} generated by posterior predictive probabilities in (7) with $\pi(t_{\text{end}})$. Training data $\mathbf{y}$ is shown in grey, while the red trace is the sequence of recommendations $\mathbf{y}_{\text{rec}}$ from sampling of the online predictive posterior after each update.

3. Experiments

We demonstrate our algorithm in a simulation study with an item space of 300 home-furnishing products.

Constructing activities As basis for the mixture model, we organise the space of items according to three different activity types, represented by Markov chains.

- $A^{(1)}$—a *functional* context, with three groups of 100 items each, of (i) kitchen items, (ii) living-room items, and (iii) bathroom items. We construct the transition matrix by assigning a number $\epsilon > 1$ to pairs of items belonging to the same group, and $1/\epsilon$ to pairs of different groups. We set the diagonal to zero, and normalise the transition matrix.
- $A^{(2)}$—an *inspirational* activity. This activity is based on inspirational product-images associated with the

items. The selection, design and composition of items in each image are made by home-furnishing specialists for marketing purposes. We represent each item by an embedding vector **x**, extracted from the prediction(s) at the last layer of a VGG-16 network pre-trained to ImageNet (we use an average from all images associated with a specific item). We then construct the transition matrix by computing

$$A_{i,j} = \frac{\exp(-||\mathbf{x}_i - \mathbf{x}_j||^2)}{\sum_{k \neq i} \exp(-||\mathbf{x}_i - \mathbf{x}_k||^2)} \tag{8}$$

for each pair $i \neq j$ of items. We set diagonal elements, $A_{i,i}$, to zero.

- $A^{(3)}$—an *add-to-cart* activity. We estimate the transition matrix (by maximum likelihood) from add-to-cart data. The shopping cart from a session is a sequence of items, with the *add-to-cart* order preserved.

Visualisation To visualise an activity, we use the t-distributed stochastic neighbour embedding of Maaten & Hinton (2008), but we skip the conversion step of a (high-dimensional) vector space to conditional probabilities, since we have these directly from the activity's transition matrix.[5] As an example, Figure 2 shows a visualization in 2D of the functional activity.

The functional shopper To emulate a user who picks items according to the functional activity, we generate a sequence of 100 items **y** from the Markov chain $A^{(1)}$. The sequence is illustrated in Figure 2 by the corresponding points of the (functional) t-SNE embedding.

For a flat, uninformative prior over the user's mixture variable, we assume equal weights, $\pi(0) = (1/3, 1/3, 1/3)$. We feed the online algorithm with data from **y**, with two items at a time for each update. The resulting posterior probabilities $\pi(t)$ are illustrated in Figure 1 as a function of the number of online updates. The posterior weights stabilises after just a few updates (~ 4, i.e. conditioned on a sequence of ~ 8 observed items) with a probability around 0.80 of the generating functional activity.

The final posterior predictive distribution, after the update with the last item in **y**, is given by $p(y|\mathbf{y})$. We use this to compute a (one-step) transition matrix A_{user} from (7) that represents the user's *individual* activity. Figure 3 shows the visualization of this activity based on A_{user}. Compared with the generating activity visualised with the three clusters in Figure 2, the user activity has (visually) concentrated to one cluster around the observed data **y** used for learning.

After each posterior update, we also draw item recommendations from the online predictive posterior and collect them in $\mathbf{y}_{\text{rec}}$. These are visualised in Figure 3, and somewhat more spread-out across the activity, than the training data.

[5]For $A^{(2)}$ this mapping is indeed given by (8).

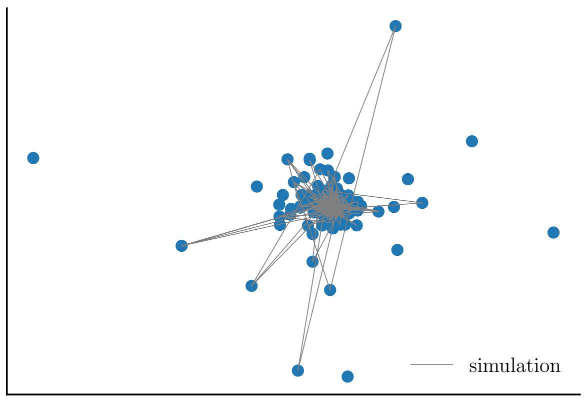

Figure 4. Visualisation (t-SNE) of the user's activity constructed from the transition matrix A_{user} generated by prior predictive probabilities in (7) with the prior $\pi(0)$. Generated (random) training data is shown in grey.

A random shopper Next we mimic a user who picks items at random: we draw 100 items according to equal probabilities over Y. The sequence is visualised in Figure 4, here with the t-SNE based on A_{user} computed with the prior mixture weights $\pi(0) = (1/3, 1/3, 1/3)$.

Posterior probabilities are shown in Figure 5. The weight for the functional activity directly drops to zero. An explanation for this is that completely random data will have a distribution 'far' from the clustered functional activity. Meanwhile, the add-to-cart and inspirational activities keep mixing whit the online updates: there is no clear stationary behaviour in the mixture distribution when trained on random data, as opposed to the posterior probabilities in Figure 1.

4. Conclusion

We have proposed a probabilistic model for recommendations. The model is a distribution over the state space of all available items, formulated as a mixture of Markov chains. Each Markov chain is constructed to represent an activity, a set of relationships between all item. The intuition behind the model is that a user picks items according these activities, or rather, a continuously changing blend of them. We proposed an online algorithm for inferring the posterior over this mixture (and thus the posterior over the items). We outlined equations for the predictive posterior and suggested a simple recommendation based on posterior sampling. Finally, we demonstrated our approach with a simulation study.

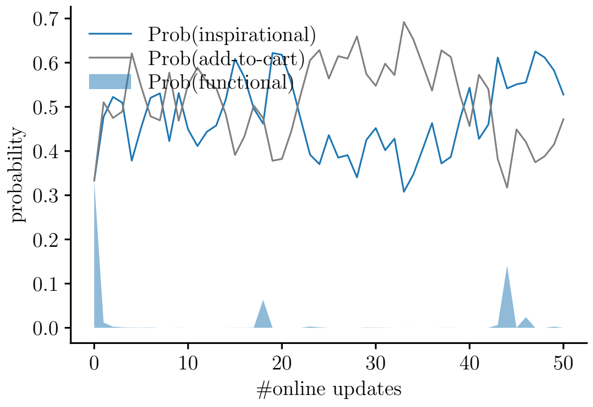

Figure 5. The random shopper. Posterior probabilities $\pi(t)$ of the categorical mixture distribution over activities, as a function of online updates.

Acknowledgements

We would like to thank Sandhya Sachidanandan, Balázs Tóth and Emil Joergensen for helpful discussions.

References

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pp. 7–10, 2016.

Hidasi, B., Karatzoglou, A., Baltrunas, L., and Tikk, D. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

Linden, G., Smith, B., and York, J. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov): 2579–2605, 2008.