2nd Symposium on Advances in Approximate Bayesian Inference, 2019 1–6

Latent Variable Session-Based Recommendation

Stephen Bonner S.A.R.BONNER@DURHAM.AC.UK
Durham University, Durham

David Rohde D.ROHDE@CRITEO.COM
Criteo AI Lab, Paris

Abstract

We present a probabilistic framework for session based recommendation. A latent variable for the user state is updated as the user views more items and we learn more about their interests. We provide computational solutions using both the re-parameterization trick and using the Bouchard bound for the softmax function, we further explore employing a variational auto-encoder and a variational Expectation-Maximization algorithm for tightening the variational bound. Finally we show that the Bouchard bound causes the denominator of the softmax to decompose into a sum enabling fast noisy gradients of the bound giving a fully probabilistic algorithm reminiscent of word2vec and a fast online EM algorithm.

1. A Latent Variable Model For Item Views

Our model describes a generative process for the types of products that user's co-view in sessions. We use u to denote a user or a session, we use t time to denote sequential time and v to denote which product they viewed from 1 to P where P is the number of products, the user's interest is described by a K dimensional latent variable $\boldsymbol{\omega}_u$ which can be interpreted as the user's interest in K topics. The session length of user u is given by T_u. We then assume the following generative process for the views in each session: $\boldsymbol{\omega}_u \sim \mathcal{N}(\mathbf{0}_K, \boldsymbol{I}_K)$, $v_{u,1}, .., v_{u,T_u} \sim \text{categorical}(\text{softmax}(\boldsymbol{\Psi}\boldsymbol{\omega}_u + \boldsymbol{\rho}))$. This model is a linear version of the model presented in Liang et al. (2018).

Consider the case where we have estimated that $\boldsymbol{\Psi}$ and $\boldsymbol{\rho}$. In production we have observed a user's online viewing history $v_{u,1}, .., v_{u,T_u}$ and we would like to produce a representation of the user's interests. Our proposal is to use Bayesian inference in order to infer $p(\boldsymbol{\omega}|v_{u,1}, .., v_{u,T_u}, \boldsymbol{\Psi}, \boldsymbol{\rho})$ as a representation of their interests. This representation of interests can then be used as a feature for training a recommender system. If we have a recommender system that has just seven products and the products have embeddings:

$$\boldsymbol{\Psi} = \begin{bmatrix} \boldsymbol{\Psi}_{\text{Sleek Phone}} \\ \boldsymbol{\Psi}_{\text{City Phone}} \\ \boldsymbol{\Psi}_{\text{Couscous}} \\ \boldsymbol{\Psi}_{\text{Rice}} \\ \boldsymbol{\Psi}_{\text{Beer}} \\ \boldsymbol{\Psi}_{\text{Female Shirt}} \\ \boldsymbol{\Psi}_{\text{Male Shirt}} \end{bmatrix} = \begin{bmatrix} .9 & 0.05 & 0 & 0.05 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & .95 & 0 & 0.1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0.2 & .7 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}, \qquad \boldsymbol{\rho} = \mathbf{0}.$$

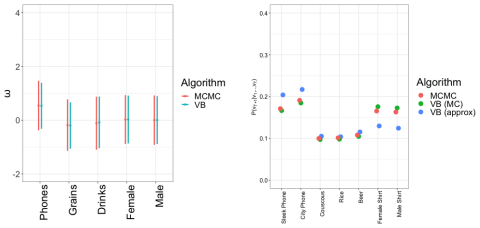

Figure 1: User representation (left) and next item prediction for a user with one sleek phone in their history

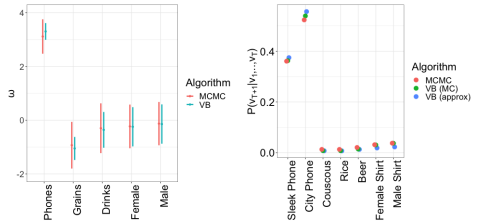

Figure 2: User representation (left) and next item prediction for a user with one sleek phone and twenty city phones in their history

We now consider how different user histories affect $p(\boldsymbol{\omega}|v_{u,1},..,v_{u,T_u},\boldsymbol{\Psi},\boldsymbol{\rho})$. Approximation of this quantity can be made accurately and easily using the Stan probabilistic programming language Stan Development Team (2018) or using variational approximations.

In Figure 1 - 2 the intuitive behavior of this simple model is demonstrated. The results of the three approximate methods are presented and shown to be in good agreement. A single product view indicates interest in that class of products, but considerable uncertainty remains. Many product views in the same class represent high certainty that the user is interested in that class. For next item prediction we consider both taking the plug-in predictive based on the posterior mean VB (approx) and using Monte Carlo samples to approximate the true predictive distribution MCMC and VB (MC).

2. Approximate Inference

2.1. Product Embedding Learning

The model we introduce has the form:

$$\log\ p(v_1,..,v_T,\boldsymbol{\omega}_u|\boldsymbol{\Psi}) = \left(\sum_t^T \boldsymbol{\Psi}_{v_t}\boldsymbol{\omega}_u + \boldsymbol{\rho}_{v_t}\right) - T\log\{\sum_p^P \exp(\boldsymbol{\Psi}_p\boldsymbol{\omega}_u + \boldsymbol{\rho}_p)\} - \frac{K}{2}\log(2\pi) - \frac{1}{2}\boldsymbol{\omega}_u^T\boldsymbol{\omega}_u.$$

If we use a normal distribution $\boldsymbol{\omega} \sim \mathcal{N}(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$, then variational bound has the form:

$$\mathcal{L} = \mathop{\mathbb{E}}_{q(\boldsymbol{\omega})}[\log\ p(v_1, .., v_T, \boldsymbol{\omega}_u|\boldsymbol{\Psi}) - \log q(\boldsymbol{\omega})] = \left(\sum_t^T \boldsymbol{\Psi}_{v_t}\boldsymbol{\mu}_q + \boldsymbol{\rho}_{v_t}\right) - T \mathop{\mathbb{E}}_{q(\boldsymbol{\omega})}[\log\{\sum_p^P \exp(\boldsymbol{\Psi}_p\boldsymbol{\omega}_u + \boldsymbol{\rho}_p)\}]$$
$$- \frac{K}{2}\log(2\pi) - \frac{1}{2}\{\boldsymbol{\mu}_q^T\boldsymbol{\mu}_q + \text{trace}(\boldsymbol{\Sigma}_q)\} + \frac{1}{2}\log|2\pi e\boldsymbol{\Sigma}_q|,$$

but we still need to be able to integrate under the denominator of the softmax.

The Bouchard bound Bouchard (2007) introduces a further approximation and additional variational parameters a, ξ but produces an analytical bound:

$$\mathcal{L} \geq \mathcal{L}_{\text{Bouch}} = \left(\sum_t^T \boldsymbol{\Psi}_{v_t}\boldsymbol{\mu}_q + \boldsymbol{\rho}_{v_t}\right) - T[a + \sum_p^P \frac{\boldsymbol{\Psi}_p\boldsymbol{\mu}_q + \boldsymbol{\rho}_p - a - \xi_p}{2}$$
$$+ \lambda_{\text{JJ}}(\xi_p)\{(\boldsymbol{\Psi}_p\boldsymbol{\mu}_q + \boldsymbol{\rho}_p - a)^2 + \boldsymbol{\Psi}_p\boldsymbol{\Sigma}_q\boldsymbol{\Psi}_p^T - \xi_p^2\} + \log(1 + e^{\xi_p})]$$
$$- \frac{K}{2}\log(2\pi) - \frac{1}{2}\{\boldsymbol{\mu}_q^T\boldsymbol{\mu}_q + \text{trace}(\boldsymbol{\Sigma}_q)\} + \frac{1}{2}\log|2\pi e\boldsymbol{\Sigma}_q|.$$

Where $\lambda_{\text{JJ}}(\cdot)$ is the Jaakola and Jordan function Jaakkola and Jordan (1997): $\lambda_{\text{JJ}}(\xi) = \frac{1}{2\xi}\left(\frac{1}{1+e^{-\xi}} - \frac{1}{2}\right)$.

Alternatively the re-parameterization trick Kingma and Welling (2014) proceeds by simulating: $\boldsymbol{\epsilon}^{(s)} \sim \mathcal{N}(\mathbf{0}_K, \boldsymbol{I}_K)$, and then computing: $\boldsymbol{\omega}^{(s)} = L_{\boldsymbol{\Sigma}_q}\boldsymbol{\epsilon}^{(s)} + \boldsymbol{\mu}_q$. Where $\boldsymbol{L}_{\boldsymbol{\Sigma}_q}\boldsymbol{L}_{\boldsymbol{\Sigma}_q}^T = \boldsymbol{\Sigma}_q$, and then optimizing the noisy lower bound:

$$\mathcal{L}_{MC} = \left(\sum_t^T \boldsymbol{\Psi}_{v_t}\boldsymbol{\mu}_q + \boldsymbol{\rho}_{v_t}\right) - T\log[\sum_p^P \exp\{\boldsymbol{\Psi}_p(L_{\boldsymbol{\Sigma}_q}\boldsymbol{\epsilon}^{(s)} + \boldsymbol{\mu}_q) + \boldsymbol{\rho}_p\}]$$
$$- \frac{K}{2}\log(2\pi) - \frac{1}{2}\{\boldsymbol{\mu}_q^T\boldsymbol{\mu}_q + \text{trace}(\boldsymbol{\Sigma}_q)\} + \frac{1}{2}\log|2\pi e\boldsymbol{\Sigma}_q|.$$

In order to prevent the variational parameters growing with the amount of data we employ a variational auto-encoder. This involves using a flexible function i.e. $\boldsymbol{\mu}_q,\ \boldsymbol{\Sigma}_q = f_\Xi(v_1, ...v_T)$, or in the case of the Bouchard bound: $\boldsymbol{\mu}_q,\ \boldsymbol{\Sigma}_q,\ a, = f_\Xi^{\text{Bouch}}(v_1, ...v_T)$ and $\xi_p = h(\boldsymbol{\Psi}_p, \boldsymbol{\rho}_p, a, \boldsymbol{\Sigma}_q, \boldsymbol{\rho}_q)$. Where any function (e.g. a deep net) can be used for $f_\Xi(\cdot)$ and $f_\Xi^{\text{Bouch}}(\cdot)$. We demonstrate that our method using the RecoGym simulation environment Rohde et al. (2018). We fit the model to the training set, we then evaluate by providing the model $v_1, ..v_{T_u-1}$ events and testing the model's ability to predict v_{T_u}.

A further approximate algorithm which is useful when P is large is to note that the bound can be written as a sum that decomposes not only in data but also over the denominator of the softmax, The noisy lower bound becomes:

Train Algorithm	Online	RC@5	DCG@5
Bouch/AE	AE	0.082	0.079
Bouch/AE	EM	**0.117**	**0.130**
RT/Deep AE	AE	0.080	0.068
RT/Deep AE	EM	0.090	0.106

Table 1: Results on the test set for all approaches on the RecoGym dataset with 2000 products. For both metrics, a higher value is better.

$$\hat{\mathcal{L}}_{\text{Bouch}}(v_1,...,v_T,n_1,...n_S,\Xi,\boldsymbol{\Psi}) = \left(\sum_t^T \boldsymbol{\Psi}_{v_t}\boldsymbol{\mu}_q + \boldsymbol{\rho}_{v_t}\right)$$
$$- T[a + \frac{P}{S}\sum_{s'=1}^{S}\frac{\boldsymbol{\Psi}_{n_{s'}}\boldsymbol{\mu}_q + \boldsymbol{\rho}_{n_{s'}} - a - \xi_{n_{s'}}}{2}$$
$$+ \lambda_{\text{JJ}}(\xi_{n_{s'}})\{(\boldsymbol{\Psi}_{n_{s'}}\boldsymbol{\mu}_q + \boldsymbol{\rho}_{n_{s'}} - a)^2 + \boldsymbol{\Psi}_{n_{s'}}\boldsymbol{\Sigma}_q\boldsymbol{\Psi}_{n_{s'}}^T - \xi_{n_{s'}}^2\} + \log(1+e^{\xi_{n_{s'}}})]$$
$$- \frac{K}{2}\log(2\pi) - \frac{1}{2}\{\boldsymbol{\mu}_q^T\boldsymbol{\mu}_q + \text{trace}(\boldsymbol{\Sigma}_q)\} + \frac{1}{2}\log|2\pi e\boldsymbol{\Sigma}_q|.$$

where $v_1,...,v_T$ are the items associated with the session and $n_1,...n_S$ are $S < P$ negative items randomly sampled. Similar to the word2vec algorithm Mikolov et al. (2013) but without any non-probabilistic heuristics.

We consider two alternative methods for training the model: **Bouch/AE** - A linear variational auto-encoder using the Bouchard bound; **RT/Deep AE** - A deep auto-encoder again using the re-parameterization trick. The deep auto-encoder consists of mapping an input of size P to three linear rectifier layers of K units each. Results showing recall@5 and discounted cumulative gain at 5 are shown in Table 1.

2.2. User Embedding Learning

The EM algorithm allows an approximation to be made of $q(\boldsymbol{\omega})$ assuming $(\boldsymbol{\Psi},\boldsymbol{\rho})$ and a user history $v_1,..,v_T$ are known and can be used in place of a variational auto-encoder. The algorithm here is the *dual* of the one presented in Bouchard (2007) as we assume the embedding $\boldsymbol{\Psi}$ is fixed and $\boldsymbol{\omega}$ is updated where the algorithm they present does the opposite. The EM algorithm consists of cycling the following update equations:

$$\boldsymbol{\Sigma}_q^{-1} = I_k + 2T\sum_p \lambda_{\text{JJ}}(\xi_p)\boldsymbol{\Psi}_p^T\boldsymbol{\Psi}_p, \quad \boldsymbol{\mu}_q = \boldsymbol{\Sigma}_q\left((\sum_t^T \boldsymbol{\Psi}_{v_t}^T) - T\left[\sum_p^P\{\frac{1}{2} + 2(\boldsymbol{\rho}_p - a)\lambda_{\text{JJ}}(\xi_p)\}\boldsymbol{\Psi}_p^T\right]\right),$$

$$a = \frac{-1 + \frac{P}{2} + \sum_p 2\lambda_{\text{JJ}}(\xi_p)(\boldsymbol{\Psi}_p\boldsymbol{\mu}_q + \boldsymbol{\rho}_p)}{2\sum_p \lambda_{\text{JJ}}(\xi_p)},$$

$$\xi_p = h(\boldsymbol{\Psi}_p,\boldsymbol{\rho}_p,a,\boldsymbol{\Sigma}_q,\boldsymbol{\rho}_q) = \sqrt{\boldsymbol{\Psi}_p\boldsymbol{\Sigma}_q\boldsymbol{\Psi}_p^T + (\boldsymbol{\Psi}_p\boldsymbol{\mu}_q + \boldsymbol{\rho}_p - a)^2}.$$

We further note that the EM algorithm is (with the exception of the a variational parameter) a fixed point update (of the natural parameters) that decomposes into a sum. The terms in the sum come from the softmax in the denominator. After substituting a co-ordinate descent update of a with a gradient descent step update, then the entire fixed point update becomes a sum:

$$(\boldsymbol{\Sigma}_q^{-1})^{\text{new}} = I_k + 2\sum_p \lambda_{\text{JJ}}(h(\boldsymbol{\Psi}_p, \boldsymbol{\rho}_p, a, \boldsymbol{\Sigma}_q, \boldsymbol{\rho}_q))\boldsymbol{\Psi}_p^T\boldsymbol{\Psi}_p,$$

$$(\boldsymbol{\Sigma}_q^{-1}\boldsymbol{\mu}_q)^{\text{new}} = \left((\sum_t^T \boldsymbol{\Psi}_{v_t}^T) - T\left[\sum_p^P \{\frac{1}{2} + 2(\boldsymbol{\rho}_p - a)\lambda_{\text{JJ}}\{h(\boldsymbol{\Psi}_p, \boldsymbol{\rho}_p, a, \boldsymbol{\Sigma}_q, \boldsymbol{\rho}_q)\}\}\boldsymbol{\Psi}_p^T\right]\right)$$

$$a^{\text{new}} = a + \frac{-1 + \frac{P}{2}}{2} + \sum_p \lambda_{\text{JJ}}\{h(\boldsymbol{\Psi}_p, \boldsymbol{\rho}_p, a, \boldsymbol{\Sigma}_q, \boldsymbol{\rho}_q)\}(\boldsymbol{\Psi}_p\boldsymbol{\mu}_q + \boldsymbol{\rho}_p) - a\lambda_{\text{JJ}}\{h(\boldsymbol{\Psi}_p, \boldsymbol{\rho}_p, a, \boldsymbol{\Sigma}_q, \boldsymbol{\rho}_q)\}$$

That is the EM algorithm can be written:

$$\left((\boldsymbol{\Sigma}_q^{-1})^{\text{new}}, (\boldsymbol{\Sigma}_q^{-1}\boldsymbol{\mu}_q)^{\text{new}}, a^{\text{new}}\right) = \sum_p^P g(\boldsymbol{\Psi}_p, \boldsymbol{\rho}_p, \boldsymbol{\Sigma}_q^{-1}, \boldsymbol{\Sigma}_q^{-1}\boldsymbol{\mu}_q, a).$$

As noted in Cappé and Moulines (2009) when an EM algorithm can be written as a fixed point update over a sum, then the Robbins Monro algorithm can be applied. Allowing updates of the form (p is chosen randomly):

$$\begin{aligned}&\left((\boldsymbol{\Sigma}_q^{-1})^{(\text{s})}, (\boldsymbol{\Sigma}_q^{-1}\boldsymbol{\mu}_q)^{(\text{s})}, a^{(\text{s})}\right)\\ &= (1 - \Delta_s)\left((\boldsymbol{\Sigma}_q^{-1})^{(\text{s}-1)}, (\boldsymbol{\Sigma}_q^{-1}\boldsymbol{\mu}_q)^{(\text{s}-1)}, a^{(\text{s}-1)}\right) + \Delta_s g(\boldsymbol{\Psi}_p, \boldsymbol{\rho}_p, (\boldsymbol{\Sigma}_q^{-1})^{(\text{s}-1)}, (\boldsymbol{\Sigma}_q^{-1}\boldsymbol{\mu}_q)^{(\text{s}-1)}, a^{(\text{s}-1)}).\end{aligned}$$

where Δ is a slowly decaying Robbins Monro sequence (Robbins and Monro (1951)) with $\Delta_1 = 1$ (meaning no initial value of $(\boldsymbol{\Sigma}_q^{-1})^{(0)}, (\boldsymbol{\Sigma}_q^{-1}\boldsymbol{\mu}_q)^{(0)}, a^{(0)}$ is needed. For large P this algorithm is many times faster than the generic EM algorithm.

What is distinct about *both* this online EM algorithm *and* the negative sampling SGD approach is that it is the denominator of the softmax that may be sub-sampled rather than individual records. The Bouchard bound is also used for decomposing the softmax into a sum in Titsias (2016) but they do per-batch optimization of the variational parameters, instead we use an auto-encoder allowing direct SGD. Our method also differs from Ruiz et al. (2018) again in using an auto-encoder allowing the more flexible SGD algorithm in place of stochastic variational inference (Hoffman et al. (2013)) which requires complete data exponential family assumptions. Finally unlike those methods we are considering variational inference of a latent variable model as well as using variational bounds to approximate the softmax.

References

Guillaume Bouchard. Efficient bounds for the softmax function, applications to inference in hybrid models, 2007.

Olivier Cappé and Eric Moulines. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613, 2009.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Tommi Jaakkola and Michael Jordan. A variational approach to bayesian logistic regression models and their extensions. In *Sixth International Workshop on Artificial Intelligence and Statistics*, volume 82, page 4, 1997.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL https://openreview.net/group?id=ICLR.cc/2014.

Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In Pierre-Antoine Champin, Fabien L. Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis, editors, *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 689–698. ACM, 2018. doi: 10.1145/3178876.3186150. URL https://doi.org/10.1145/3178876.3186150.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, 22(3):400–407, 1951.

David Rohde, Stephen Bonner, Travis Dunlop, Flavian Vasile, and Alexandros Karatzoglou. Recogym: A reinforcement learning environment for the problem of product recommendation in online advertising. In *REVEAL workshop, ACM Conference on Recommender Systems 2018*, 2018.

Francisco JR Ruiz, Michalis K Titsias, Adji B Dieng, and David M Blei. Augment and reduce: Stochastic inference for large categorical distributions. *arXiv preprint arXiv:1802.04220*, 2018.

Stan Development Team. Pystan: the python interface to stan, version 2.17.1.0., 2018.

Michalis Titsias. One-vs-each approximation to softmax for scalable estimation of probabilities. In *Advances in Neural Information Processing Systems*, pages 4161–4169, 2016.