Bayesian Statistical Methods - 20231
Final Project

Bahcivanoglu, Efecan
efecan.bahcivanoglu@studbocconi.it

January 7, 2024

Contents

1 Introduction

In this paper, I'll compare Hierarchical Poisson Factorization to Bayesian Nonparametric Poisson Factorization in Recommender Systems. In order to do that, I'll need to explain further concepts such as variational inference and stochastic variational inference. Last but not least, I'll use Variational Inference for Hierarchical Poisson Factorization, implementing Coordinate Ascent Variational Inference Algorithm and check our results on a real dataset.

2 Recommender Systems

Over the last twenty years consuming goods became significantly diversified with new features claimed to address certain customer segments. Thus, modern consumers are faced with many choices, whether ordering an item from Amazon, listening to a song on Spotify, or watching a movie on Netflix. To make this process less cognitively costly various recommender algorithms have been created.

One of the most successful approaches is based on matrix factorization. Simply, a latent vector which contains user preferences is created for each user, for each item same vector is created. Then, the dot product is taken to estimate the user's rating of item i. Of course, once the latent factors are known, it's easy to produce estimates. The real problem, though how to estimate the latent factors.

3 Posterior Inference

Bayesian statistics offers a probabilistic approach to inference, where uncertainty in model parameters is expressed in terms of probability distributions. There are various approaches to finding the posterior. We can derive the posterior analytically if we have conjugacy, Beta-Binomial, Poisson-Gamma or Multinomial Dirichlet. Unfortunately, in most cases, we cannot. Monte Carlo Markov Chain algorithms are used when direct sampling from a probability distribution is difficult. The idea is to construct a Markov Chain that has the posterior distribution as its equilibrium distribution. Markov Chain is run for a number of steps, and then the states of the chain are used as samples from the distribution. An initial burn-in period is required for the chain, and convergence has to be established. Even though they provide a direct sampling from the posterior distribution, they are computationally intensive. Here, our focus will lie on Variational Inference and Stochastic Variational Inference, as they are the two algorithms used in our model.

3.1 Variational Inference

In Variational Inference, we use an auxiliary distribution. The idea is that we choose a family of distribution. Our goal is to vary that distribution so that the Kullback Leibler divergence between that distribution and the posterior is as small as possible. Assuming Z are the latent variables and D is the data:

$$q(Z) = \text{argmin}\, \mathcal{KL}(q(Z) \mid\mid p(Z|X=D)) \quad \textit{where} \;\; q(Z) \in Q \tag{3.1.1}$$

Bayes Rule

$$p(Z|X=D) = \frac{p(X=D|Z)p(Z)}{p(X=D)} \tag{3.1.2}$$

Problem with the marginal normalizing constant.

$$p(X) = \int_{Z_0} ... \int_{Z_{D-1}} p(X,Z)\; dZ_0...dZ_1 \tag{3.1.3}$$

Instead of this, we try to approximate the posterior q(Z) using a family of simple distributions Q.

$$\mathcal{KL}(q(z) \mid\mid p(Z|X=D)) = \mathbb{E}_{q(Z)}\left[\log \frac{q(Z)}{p(Z|X=D)}\right] \tag{3.1.4}$$

Where subscript of the expectation means the expectation is taken over set z distributed according to q(Z).

$$=> \int_{Z_0} ... \int_{Z_{D-1}} q(Z) \log \frac{q(Z)}{p(Z|X=D)}\; dZ_0...dZ_1 \tag{3.1.5}$$

The remedy is using the Bayes rule to re-write the posterior in the denominator. $\overline{Z}$ stands for all of the latent variables.

$$\mathcal{KL}(q(Z)||p(Z|X=D)) = \mathbb{E}_{q(Z)}\left[\log\frac{q(Z)}{p(Z|X=D)}\right] \;=\; \mathbb{E}_{q(Z)}\left[\frac{q(Z)p(D)}{p(Z,X=D)}\right]$$

$$= \int_{\overline{Z}} q(\overline{Z})\log\left[\frac{q(\overline{Z})p(D)}{p(\overline{Z},X=D)}\right]d\overline{Z}$$

Finally, if we use the log property to write these terms as a summation, what we have:

$$\int_{\overline{Z}} q(\overline{Z})\log\frac{q(d\overline{Z})}{p(\overline{Z},X=D)}d\overline{Z} \;+\; \int_{\overline{Z}} q(\overline{Z})\log p(X=D)d\overline{Z} \tag{3.1.6}$$

We can write this in expectation form. If we change the numerator and the denominator in the first term and place a minus in front, we have:

$$-\mathbb{E}_{q(Z)}\left[\log\frac{p(Z,X=D)}{q(z)}\right] \;+\; \mathbb{E}_{q(Z)}\left[\log p(X=D)\right] \tag{3.1.7}$$

Since the second term is an expectation over a quantity that doesn't depend on q(Z).

$$-\mathbb{E}_{q(Z)}\left[\log\frac{p(Z,X=D)}{q(z)}\right] \;+\; \log p(X=D) \tag{3.1.8}$$

The left term is called Evidence Lower Bound (ELBO). The right term is the log probability of the data. Since probability is between 0 and 1, this term is smaller or equal to zero. Furthermore, it's a fixed quantity and doesn't change its value based on which type of surrogate distribution we use to approximate the posterior. After all these steps, our final equation becomes:

$$KL = -ELBO \;+\; \log p(X=D) \tag{3.1.9}$$

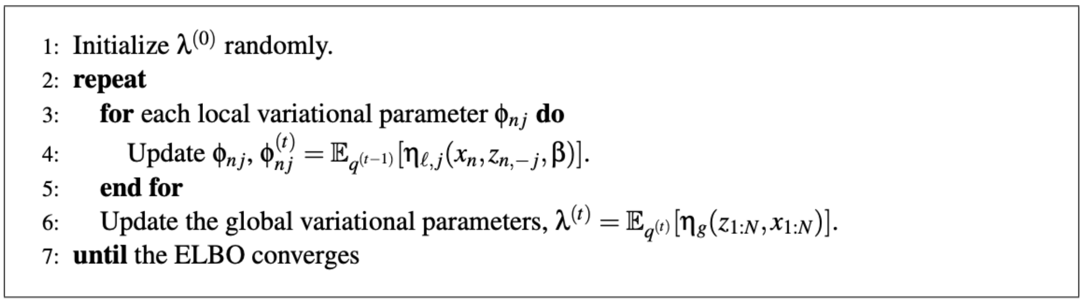
Kullback-Leibler convergence is a positive quantity unless two distributions are exactly the same; in this case, it is equal to zero. Then we expect ELBO to be greater logp(D) in absolute terms. Here, minimizing KL divergence is equal to maximizing ELBO. There's an important assumption: Mean-Field Approximation, which forces independence among B groups of parameters. Here Θ stands for the parameter of the variational distribution.

$$q(\Theta) = \prod_{b=1}^{B} q(\Theta_b), \tag{3.1.10}$$

Mean-Field forces independence among B groups of parameters while dependence is preserved within each block. In the end, our optimization problem becomes:

$$\arg\max_{q \in \mathcal{Q}} \text{ELBO}\{q(\Theta)\} \quad \text{subject to} \quad \mathcal{Q} = \left\{ q(\Theta) : q(\Theta) = \prod_{b=1}^{B} q(\Theta_b) \right\} \tag{3.1.11}$$

For a model with local and global hidden variables, we update the local parameters using all the observations and conditional on all the other variables. The fact that we have to go over all of the observations at each iteration is computationally inefficient. Stochastic Variational Inference is another method to address this inefficiency, but it's beyond the scope of this paper. Coordinate Ascent Mean Field Variational Inference is given in Hoffmann 2013 as such:



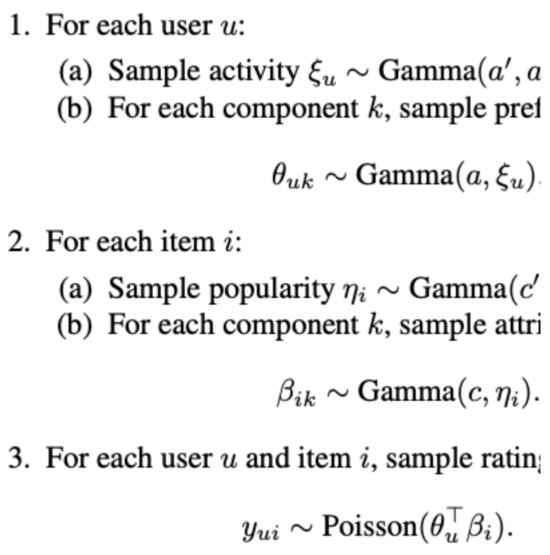
```
1: Initialize $\lambda^{(0)}$ randomly.
2: **repeat**
3:     **for** each local variational parameter $\phi_{nj}$ **do**
4:         Update $\phi_{nj}$, $\phi_{nj}^{(t)} = \mathbb{E}_{q^{(t-1)}}[\eta_{\ell,j}(x_n, z_{n,-j}, \beta)]$.
5:     **end for**
6:     Update the global variational parameters, $\lambda^{(t)} = \mathbb{E}_{q^{(t)}}[\eta_g(z_{1:N}, x_{1:N})]$.
7: **until** the ELBO converges
```

4 Hierachical Poisson Factorization

This chapter is based on Gopalan's 2015 Scalable Recommendation with Hierarchical Poisson Factorization paper. All of the code is written by Efecan Bahcivanoglu. As we established before, in matrix factorization, each item and user is represented by a vector of K latent variables. β_i represents item i's vector of K latent attributes, while θ_u represents user u's vector of K latent attributes. In this case, each latent variable is sampled from a Gamma distribution.

$$\beta_i = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \end{bmatrix} \qquad \theta_u = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \end{bmatrix} \tag{4.0.1}$$

Individual ratings are parameterized by the inner product of latent features: $y_{ui} \sim Poisson(\theta_u^T \beta_i)$. Gopalan et al. places additional priors on the user and item-specific rate parameters. In their words: "This hierarchical structure allows us to capture the diversity of users, some tending to consume more than others, and the diversity of items, some being more popular than others. The literature on recommendation systems suggests that a good model must capture such heterogeneity across users and items." Thus, the generating mechanism becomes:



1. For each user u:
   (a) Sample activity $\xi_u \sim \text{Gamma}(a', a$
   (b) For each component k, sample pref

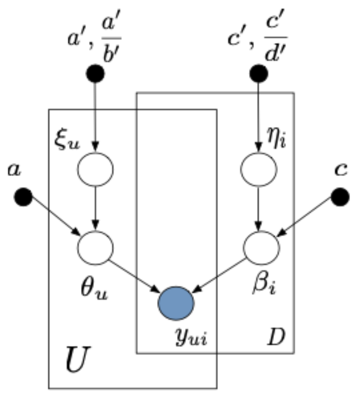
   $$\theta_{uk} \sim \text{Gamma}(a, \xi_u)$$

2. For each item i:
   (a) Sample popularity $\eta_i \sim \text{Gamma}(c'$
   (b) For each component k, sample attri

   $$\beta_{ik} \sim \text{Gamma}(c, \eta_i).$$

3. For each user u and item i, sample ratin

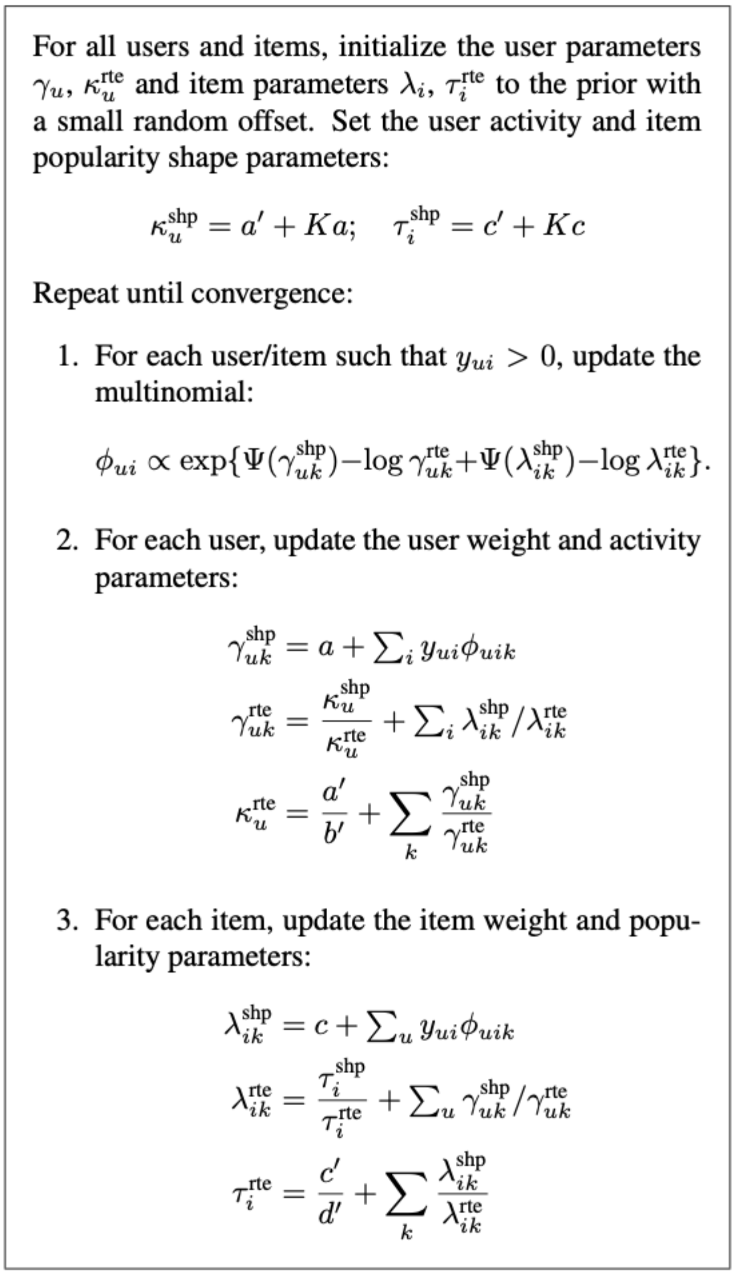
   $$y_{ui} \sim \text{Poisson}(\theta_u^\top \beta_i).$$



Our variational distribution after applying mean-field assumption can be written as:

$$q(\beta, \theta, \eta, \gamma, z) = \prod_{i,k} q(\beta_{ik}|\lambda_{ik}) \prod_{u,k} q(\theta_{uk}|\gamma_{uk}) \prod_{u} q(\xi_u|\kappa_u) \prod_{i} q(\eta_i|\tau_i) \prod_{u,i} q(z_{ui}|\rho_{ui}) \tag{4.0.2}$$

Coordinate Ascent Algorithm:



For all users and items, initialize the user parameters γ_u, κ_u^{rte} and item parameters λ_i, τ_i^{rte} to the prior with a small random offset. Set the user activity and item popularity shape parameters:

$$\kappa_u^{\text{shp}} = a' + Ka; \quad \tau_i^{\text{shp}} = c' + Kc$$

Repeat until convergence:

1. For each user/item such that $y_{ui} > 0$, update the multinomial:

$$\phi_{ui} \propto \exp\{\Psi(\gamma_{uk}^{\text{shp}}) - \log \gamma_{uk}^{\text{rte}} + \Psi(\lambda_{ik}^{\text{shp}}) - \log \lambda_{ik}^{\text{rte}}\}.$$

2. For each user, update the user weight and activity parameters:

$$\begin{aligned}
\gamma_{uk}^{\text{shp}} &= a + \textstyle\sum_i y_{ui}\phi_{uik} \\
\gamma_{uk}^{\text{rte}} &= \frac{\kappa_u^{\text{shp}}}{\kappa_u^{\text{rte}}} + \textstyle\sum_i \lambda_{ik}^{\text{shp}} / \lambda_{ik}^{\text{rte}} \\
\kappa_u^{\text{rte}} &= \frac{a'}{b'} + \sum_k \frac{\gamma_{uk}^{\text{shp}}}{\gamma_{uk}^{\text{rte}}}
\end{aligned}$$

3. For each item, update the item weight and popularity parameters:

$$\begin{aligned}
\lambda_{ik}^{\text{shp}} &= c + \textstyle\sum_u y_{ui}\phi_{uik} \\
\lambda_{ik}^{\text{rte}} &= \frac{\tau_i^{\text{shp}}}{\tau_i^{\text{rte}}} + \textstyle\sum_u \gamma_{uk}^{\text{shp}} / \gamma_{uk}^{\text{rte}} \\
\tau_i^{\text{rte}} &= \frac{c'}{d'} + \sum_k \frac{\lambda_{ik}^{\text{shp}}}{\lambda_{ik}^{\text{rte}}}
\end{aligned}$$

MovieLens 10m dataset is used for empirical study. It contains 10 million ratings applied to 10,000 movies by 72,000 users. Furthermore, in the Data Preprocess notebook, sampling based on users is carried out, which creates 10 subsamples at the end. In the CAVI notebook, one of the subsamples is further divided into training and validation sets. Following the paper's guidance, convergence is measured by computing the prediction accuracy on the validation set, stopping when the change in log-likelihood is less than 0.0001%

```python
def initiliaze_parameters(num_users, num_items, k, a=0.3, a_prime=0.3, c=0.3, c_prime=0.3, b_prime=1.0, d_prime=1.0):

    k_shp = (a_prime + k * a) * np.ones(num_users)
    t_shp = (c_prime + k * c) * np.ones(num_items)
    k_rte = 1 * np.ones(num_users)
    t_rte = 1 * np.ones(num_items)


    gamma_rte = (a_prime + 0.01) * np.ones((num_users, k))
    gamma_shp = a * np.ones((num_users, k))
    lambda_rte = (c_prime + 0.01) * np.ones((num_items, k))
    lambda_shp = c * np.ones((num_items, k))

    phi = np.empty((num_users, num_items, k))

    return gamma_shp, gamma_rte, lambda_shp, lambda_rte, k_shp, k_rte, t_shp, t_rte, phi, a, a_prime, c, c_prime, b_prime, d_prime
```

Furthermore, after initializing the parameters, updates are given according to.

```python
def CAVI(num_users, num_items, k, y_ui, gamma_shp, gamma_rte, lambda_shp, lambda_rte, k_shp, k_rte, t_shp, t_rte, phi):
    for u in range(num_users):
        for i in range(num_items):
            if y_ui[u, i] > 0:
                exponent = (digamma(gamma_shp[u,:]) - np.log(gamma_rte[u,:]) +
                            digamma(lambda_shp[i,:]) - np.log(lambda_rte[i,:]))
                phi[u, i, :] = np.exp(exponent) / np.sum(np.exp(
exponent))

    for u in range(num_users):
        for d in range(k):
            k_sum = y_ui[u, :] * phi[u,:,d]
            gamma_shp[u,d] = a + np.sum(k_sum)

    for u in range(num_users):
        for d in range(k):
            k_sum = lambda_shp[:,d]/lambda_rte[:,d]
            gamma_rte[u,d] = k_shp[u] / k_rte[u] + np.sum(k_sum)

    for u in range(num_users):
        k_sum = 0
        for d in range(k):
            k_sum += gamma_shp[u,d] / gamma_rte[u,d]
        k_rte[u] = a_prime / b_prime + k_sum

    for i in range(num_items):
        for d in range(k):
            k_sum = y_ui[:, i] * phi[:,i,d]
            lambda_shp[i,d] = c + np.sum(k_sum)

    for i in range(num_items):
        for d in range(k):
            k_sum = gamma_shp[:,d]/gamma_rte[:,d]
            lambda_rte[i,d] = t_shp[i] / t_rte[i] + np.sum(k_sum)

    for i in range(num_items):
        k_sum = 0
        for d in range(k):
            k_sum += lambda_shp[i,d] / lambda_rte[i,d]
        t_rte[i] = c_prime / d_prime + k_sum

    return gamma_shp, gamma_rte, lambda_shp, lambda_rte, k_shp,
k_rte, t_shp, t_rte, phi
```

Predictions are generated in the notebook. Since the expected value of Poisson is the parameter itself, the dot product of the latent features yields the rating.

All the images are taken from Blei 2013 and Gopalan 2015.

References

[1] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. *Variational inference: A review for statisticians*. Journal of the American Statistical Association, 112(518):859–877, 2017.

[2] Matthew D. Hoffman et al. *Stochastic variational inference*. Journal of Machine Learning Research, 2013.

[3] Prem Gopalan, Jake M. Hofman, and David M. Blei. *Scalable Recommendation with Hierarchical Poisson Factorization*. UAI, 2015.

[4] GroupLens (2009). MovieLens 10M Dataset. http://grouplens.org/datasets/movielens/.