Supplementary Material for the Paper: Hierarchical Negative Binomial Factorization for Recommender Systems on Implicit Feedback

Complete Conditionals of Failure Exposure Count and Failure Exposure Factors

Complete conditional of user exposure count. According to the generative process of HNBF, failure exposure count (FEC) r_{ui} does not follow the relationship of conjugate prior. Even though, we can obtain the closed form of the complete conditional from the definition of ELBO. We derive the complete conditional of r_{ui} as follows.

Given joint probability $\mathrm{Ga}(d_{ui}; r_{ui}, r_{ui})\mathrm{Ga}(r_{ui}; g^+, \frac{g^+}{h^+})$, according to the evidence lower bound (ELBO), we have the conditional

$$\begin{aligned}\mathcal{L} =&\mathbb{E}_q[\log p(r_{ui}|d_{ui}, g^+, f^+)] - \mathbb{E}_q[\log q(r_{ui}|r_{ui}^{shp}, r_{ui}^{rte})]\\ =&\mathbb{E}_q[\log \mathrm{Ga}(d_{ui}; r_{ui}, r_{ui})] + \mathbb{E}_q[\log \mathrm{Ga}(r_{ui}; g^+, \frac{g^+}{h^+})]-\\ &\mathbb{E}_q[\log \mathrm{Ga}(r_{ui}; r_{ui}^{shp}, r_{ui}^{rte})] + \mathrm{const}.\end{aligned}$$

To derive the coordinate ascent update, we take the gradient

$$\begin{aligned}\nabla_{r_{ui}}\mathcal{L} =&\mathbb{E}_q[\log(r_{ui}) - \Psi(r_{ui}) + \frac{g^+ - 1}{r_{ui}} - \frac{r_{ui}^{shp} - 1}{r_{ui}}]+\\ &1 + \log(d_{ui}) - d_{ui} - \frac{g^+}{h^+} + r_{ui}^{rte}.\end{aligned}$$

We can let this gradient be zero by setting

$$\begin{aligned}r_{ui}^{shp} &= g^+ + r_{ui}(\log(r_{ui}) - \Psi(r_{ui}),\\ r_{ui}^{rte} &= \frac{g^+}{h^+} + d_{ui} - \log(d_{ui}) - 1.\end{aligned}$$

Complete conditional of failure exposure factors. The complete conditional of failure exposure factors μ_u and π_i can also derived by the similar mean. We will show the derivation with respect to μ_u for example as follows.

Given joint probability

$$\prod_k \mathrm{Ga}(\gamma_{uk}; \mu_u, \mu_u)\mathrm{Ga}(\mu_u; g^0, \frac{g^0}{h^0}),$$

according to the evidence lower bound (ELBO), we have the conditional

$$\begin{aligned}\mathcal{L} =&\mathbb{E}_q[\sum_{k=1}^K \log p(\mu_u|\gamma_{uk}, g^0, h^0)] - \mathbb{E}_q[\log q(\mu_u|\mu_u^{shp}, \mu_u^{rte})]\\ =&\mathbb{E}_q[\sum_{k=1}^K \log \mathrm{Ga}(\gamma_{uk}; \mu_u, \mu_u)] + \mathbb{E}_q[\log \mathrm{Ga}(\mu_u; g^0, \frac{g^0}{h^0})]-\\ &\mathbb{E}_q[\log \mathrm{Ga}(\mu_u; \mu_u^{shp}, \mu_u^{rte})] + \mathrm{const}.\end{aligned}$$

To derive the coordinate ascent update, we take the gradient

$$\begin{aligned}\nabla_{r_{ui}}\mathcal{L} =&\mathbb{E}_q[K\log\mu_u - K\Psi(\mu_u) + \frac{g^0 - 1}{\mu_u} - \frac{\mu_u^{shp} - 1}{\mu_u}]+\\ &K + \sum_{k=1}^K \log\gamma_{uk} - \sum_{k=1}^K \gamma_{uk} - \frac{g^0}{h^0} + \mu_u^{rte}.\end{aligned}$$

We can let this gradient be zero by setting

$$\begin{aligned}\mu_u^{shp} &= g^0 + K\mu_u(\log\mu_u - \Psi(\mu_u)),\\ \mu_u^{rte} &= \frac{g^0}{h^0} + \sum_{k=1}^K \gamma_{uk} - \sum_{k=1}^K \log(\gamma_{uk}) - K.\end{aligned}$$

Complete Conditionals of Dispersion and Dispersion Factors

Complete conditional of dispersion. Considering the conditional $p(d_{ui}|x_{ui}, \boldsymbol{\theta}_u^\top\boldsymbol{\beta}_i, r_{ui})$, we have the log posteriori

$$\begin{aligned}\mathcal{L} =&\mathbb{E}_q[\log p(d_{ui}|x_{ui}, \boldsymbol{\theta}_u^\top\boldsymbol{\beta}_i, r_{ui})] - \mathbb{E}_q[\log q(d_{ui}|d_{ui}^{shp}, d_{ui}^{rte})]\\ =&\mathbb{E}_q[\log \mathrm{Pois}(x_{ui}; d_{ui}\boldsymbol{\theta}_u^\top\boldsymbol{\beta}_i)] + \mathbb{E}_q[\log \mathrm{Ga}(d_{ui}; r_{ui}, r_{ui})]-\\ &\mathbb{E}_q[\log \mathrm{Ga}(d_{ui}; d_{ui}^{shp}, d_{ui}^{rte})] + \mathrm{const}.\end{aligned}$$

We can let the gradient of $\mathcal{L}$ be zero by setting

$$\begin{aligned}d_{ui}^{shp} &= r_{ui} + x_{ui},\\ d_{ui}^{rte} &= r_{ui} + \boldsymbol{\theta}_u^\top\boldsymbol{\beta}_i.\end{aligned}$$

Complete conditional of dispersion factors. The dispersion factors γ_{uk} and δ_{ik} can also be derived by the similar mean. We will show the derivation of γ_{uk} for example. Notice that γ_{uk} and δ_{ik} are the auxiliary variables. Similar to $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$, we can regard the inner product of the two dispersion factors as a K-vector of Poisson counts that sum to the

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

value which multiplied by $\frac{1}{K}\boldsymbol{\theta}_u^\top \boldsymbol{\beta}_i$ is equal to observation x_{ui}. Notice that dispersion factors are used to estimate the dispersion of zero entries so that observation x_{ui} is equal to zero. Considering the log posteriori

$$\begin{aligned}\mathcal{L} =&\mathbb{E}_q[\sum_{i\in\mathcal{I}_u^0}\log p(\gamma_{uk}|0,\delta_{ik},\boldsymbol{\theta}_u^\top\boldsymbol{\beta}_i,\mu_u)]-\\ &\mathbb{E}_q[\log q(\gamma_{uk}|\gamma_{uk}^{shp},\gamma_{uk}^{rte})]\\ =&\mathbb{E}_q[\sum_{i\in\mathcal{I}_u^0}\log \mathrm{Pois}(0;\frac{1}{K}\gamma_{uk}\delta_{ik}\boldsymbol{\theta}_u^\top\boldsymbol{\beta}_i)]+\\ &\mathbb{E}_q[\log \mathrm{Ga}(\gamma_{uk};\mu_u,\mu_u)]-\\ &\mathbb{E}_q[\log \mathrm{Ga}(\gamma_{uk};\gamma_{uk}^{shp},\gamma_{uk}^{rte})]+\mathrm{const.}\end{aligned}$$

To derive the coordinate ascent update, we take the gradient,

$$\begin{aligned}\nabla_{\gamma_{uk}}\mathcal{L} =&\mathbb{E}_q[-\frac{1}{K}\sum_{i\in\mathcal{I}_u^0}\delta_{ik}\boldsymbol{\theta}_u^\top\boldsymbol{\beta}_i]+\mathbb{E}_q[\frac{\mu_u-1}{\gamma_{uk}}-\mu_u]-\\ &\mathbb{E}_q[\frac{\gamma_{uk}^{shp}-1}{\gamma_{uk}}-\gamma_{uk}^{rte}]\end{aligned}$$

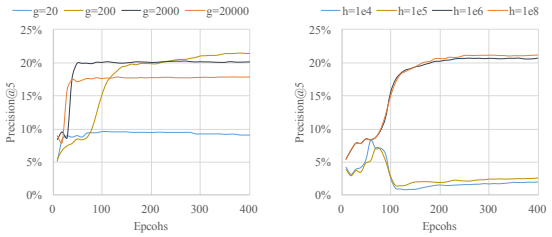
We can let the gradient of $\mathcal{L}$ be zero by setting

$$\begin{aligned}\gamma_{uk}^{shp}&=\mu_u,\\ \gamma_{uk}^{rte}&=\mu_u+\frac{1}{K}\sum_{i\in\mathcal{I}_u^0}\delta_{ik}\boldsymbol{\theta}_u^\top\boldsymbol{\beta}_i.\end{aligned}$$

Hyperparameter Setting for Dispersion Variables

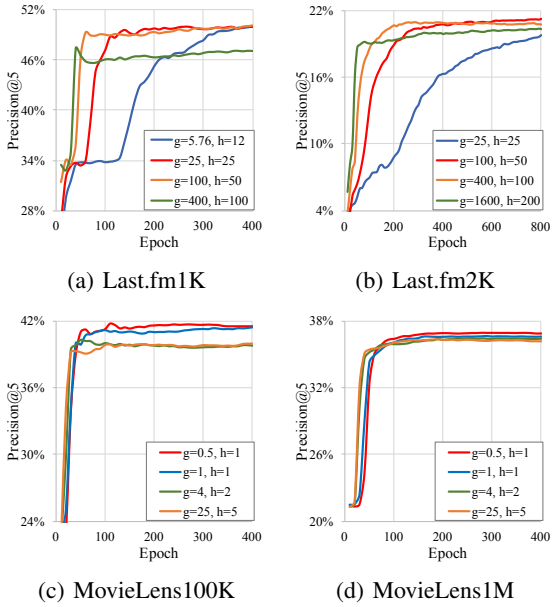
The convergence of HNBF with respect to g and h. Like previous works that consider data dispersion (Basbug and Engelhardt 2017; Gouvert, Oberlin, and Févotte 2018), the setting of the prior parameters of dispersion strongly affects the performance. Figure 1 (a) shows the convergence curve with different settings of prior g when $K = 20$. When g is small, both r_{ui} and d_{ui} variate freely. As such, $p(x_{ui}|d_{ui}\boldsymbol{\theta}_u^\top\boldsymbol{\beta}_i)$ converges fast, which makes $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ be updated slowly. By contrast, when g is vast, r_{ui} will be fixed to h. Thus, HNBF with fixed r_{ui} will reduce to NBF since the prior of a dispersion variable r_{ui} is set to a constant in NBF. In this case, HNBF converges fast but fails to reach a satisfying performance. On the other hand, we conduct the experiment with different settings of h^+, as shown in Figure 1 (b). When h^+ is small, the zero dispersion model will endure strong perturbations during updating the inference model, thus causing bad performance. Hence, it is recommended to set h^+ in no small number.

Hyperparameter selection. In the prior of the rate parameter of gamma variables (i.e., $\frac{b}{c}$, $\frac{g^+}{h^+}$, and $\frac{g^0}{h^0}$) and their denominators (i.e., f, h^+, and h^0) control the mean of the corresponding variables because, for example, the prior mean of failure exposure count $r_{ui} \sim \mathrm{Ga}(g^+, g^+/h^+)$ is $\frac{g^+}{g^+/h^+} = h$ and its variance is $\frac{g^+}{(g^+/h^+)^2} = (h^+)^2/g^+$. With a fixed mean, the prior of the rate parameter of gamma variables control the variance. Figure 2 shows the performance of FastHNBF with different prior parameters, i.e., g^+ and h^+, setting on four datasets: Last.fm1K, Last.fm2K, Movielens100K and Movielens1M, where the prior parameters of the inference model are set $(a, b, c) = (3, 1, 0.1)$ for implicit count data (i.e., Last.fm1L and Last.fm2K), and $(a, b, c) = (0.3, 0.1, 1)$ for explicit rating data (i.e., MovieLens100K and MovieLens1M), respectively.



(a) Prec@5 with g^+ (b) Prec@5 with h^+

Figure 1: The convergence with various setting on Last.fm2K.



(a) Last.fm1K (b) Last.fm2K

(c) MovieLens100K (d) MovieLens1M

Figure 2: The convergence with various setting of prior parameters g^+ and h^+.

In the experiment on implicit count data, i.e., Last.fm1K and Last.fm2K, we fix the prior variance at 25 (i.e., $h^2/g = 25$) and show the effect of the prior h on the performance in Figure 2 (a) and (b). Recall that h represents the prior expectation of FEC r_{ui}. With the increasing of h, the number of epochs needed for the model to converge gradually decreases. Indeed, a large value of h causes fixed dispersion d_{ui}, thus making the inference model converge faster with the cost of a decline in performance. By contrast, when h is set to a too-small value, the model would take much time to converge. In Last.fm1K, where $\mathbb{E}_{\mathcal{X}^+}[x_{ui}] = 21.3$ and $\mathrm{Var}_{\mathcal{X}^+}[x_{ui}] = 118.5$, the model performs well by setting g and h to 25. When $h = 12$, the model converges

slowly. When $h = 200$, the model converges quite fast but merely reaches 47% in terms of precision@5. In Last.fm2K, where $\mathbb{E}_{\mathcal{X}^+}[x_{ui}] = 746.3$ and $\mathrm{Var}_{\mathcal{X}^+}[x_{ui}] = 3758.5$, the model performs well when h is larger than 100. Notice that the model with prior mean $h = 25$ does not perform well and converge quite slow. As a result, it is recommended to use a value that is smaller than the mean of the values in nonzero entries, namely $h = 50$ in the experiment on these two datasets.

In Figure 2 (c) and (d), fixing the prior variance at 1, we conduct the experiment on explicit rating, i.e., MovieLens100K and MovieLens1M, and show the effect of the prior h on the performance. We also run HNBF with $(g, h) = (0.5, 1)$ to fix the variance of failure exposure count at 0.5. Since the scale of the value in an entry is limited to $[0, 5]$ in explicit rating, prior mean h should not be set to a large value; otherwise, large h, i.e., the prior mean of failure exposure count, will fix the dispersion d_{ui} at 1, thus making the proposed model reduce to HPF approximately. In these two datasets, all the models with different setting converge fast. In MovieLens100K, where $\mathbb{E}_{\mathcal{X}^+}[x_{ui}] = 3.53$ and $\mathrm{Var}_{\mathcal{X}^+}[x_{ui}] = 1.13$, the model performs well by setting g and h to 0.5 and 1, respectively. When $h = 5$, the model converges slightly faster than others but performs worse. In MovieLens1M, where $\mathbb{E}_{\mathcal{X}^+}[x_{ui}] = 3.58$ and $\mathrm{Var}_{\mathcal{X}^+}[x_{ui}] = 1.12$, the model with setting $(g, h) = (0.5, 1)$ outperforms the others. As a result, in explicit rating datasets, where the scale of entry value is limited, it is recommended to set the prior parameter (g, h) to $(0.5, 1)$.

In practice, we can set g and h to adaptive values respectively by considering the mean and variance of values in nonzero entries, namely, $\mathbb{E}_{\mathcal{X}^+}[x_{ui}]$ and $\mathrm{Var}_{\mathcal{X}^+}[x_{ui}]$. During the training phase, we want the variance of inference $\boldsymbol{\theta}_u\boldsymbol{\beta}_i$ to be large. Considering the posterior $p(\boldsymbol{\theta}_u\boldsymbol{\beta}_i|x_{ui}, r_{ui})$ which is assumed to be a NB, we have the following formula with respect to r_{ui} derived from its variance:

$$\mathbb{E}_{\mathcal{X}^+}[r_{ui}] = \frac{\mathbb{E}_{\mathcal{X}^+}[x_{ui}]^2}{\mathrm{Var}_{\mathcal{X}^+}[\boldsymbol{\theta}_u\boldsymbol{\beta}_i] - \mathbb{E}_{\mathcal{X}^+}[x_{ui}]}. \quad (1)$$

If $\mathrm{Var}_{\mathcal{X}^+}[\boldsymbol{\theta}_u\boldsymbol{\beta}_i] \approx \mathbb{E}_{\mathcal{X}^+}[x_{ui}]$, expectation $\mathbb{E}_{\mathcal{X}^+}[r_{ui}]$ will be very large. Since we priorly assume that $\mathrm{Var}_{\mathcal{X}^+}[\boldsymbol{\theta}_u\boldsymbol{\beta}_i] \gg \mathbb{E}_{\mathcal{X}^+}[x_{ui}]$, we have the expectation of r_{ui}

$$\mathbb{E}_{\mathcal{X}^+}[r_{ui}] = \frac{\mathbb{E}_{\mathcal{X}^+}[x_{ui}]^2}{\mathrm{Var}_{\mathcal{X}^+}[\boldsymbol{\theta}_u\boldsymbol{\beta}_i]}. \quad (2)$$

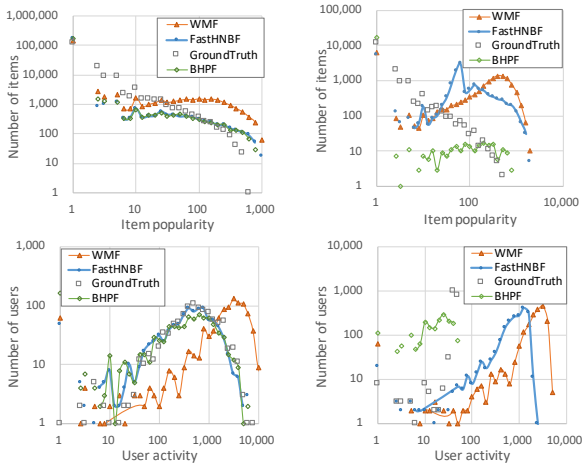
Thus, to avoid set h^+ to a small value, we empirically set

$$h^+ = \min\left(\frac{\mathbb{E}_{\mathcal{X}^+}[x_{ui}]^2}{\mathrm{Var}_{\mathcal{X}^+}[\boldsymbol{\theta}_u\boldsymbol{\beta}_i]}, \frac{\mathbb{E}_{\mathcal{X}^+}[x_{ui}]}{3}\right). \quad (3)$$

In the inference model, the prior parameters do not affect the performance much practically. In the experiment, the settings of prior parameters can be categorized into two folds, implicit count and explicit rating. In implicit count, considering the mean of values in nonzero entries is large, we set the prior parameter to $(3, 1, 0.1)$ to fix the prior mean at 10. As such, the prior mean of $\boldsymbol{\theta}_u^\top\boldsymbol{\beta}_i$ is fixed at 0.3. In explicit rating, we set the prior parameter to $(0.3, 0.1, 1)$ to fix the prior mean of μ_u and π_i at 0.1. As such, the prior mean of $\boldsymbol{\theta}_u^\top\boldsymbol{\beta}_i$ is fixed at 3.

Although the proposed method has seven hyperparameters, tuning these hyperparameters will be easy by following the policy stated as follows. Firstly, we tune the three hyperparameters in the inference model. In implicit count data, it is recommended to set (a, b, c) to $(3, 1, 0.1)$ to make the variance of latent variables in the inference model larger. In explicit rating, it is recommended to set (a, b, c) to $(0.3, 0.1, 1)$ since the mean and variance of the ground truth values in nonzero entries are limited. The dispersion model has four hyperparameters. We can categorize them into only two types: g and h. In the dispersion model for nonzero entries, we can set h^+ according to Eq. (3), and set g^+ to h^+ to limit the variance of the variables for FEC priorly. In the dispersion model for zero entries, hyperparameters (g^0, h^0) do not affect the performance when set to large adaptive values. Practically, we set (g^0, h^0) to $(10^1, 10^8)$ to run the proposed model on smaller datasets such as Last.fm1K, Last.fm2K, MovieLens100K and MovieLens1M. The proposed model with (g^0, h^0) set to $(10^2, 10^{12})$ can perform well on large datasets, e.g., Last.fm360K and MovieLens20M. In the dispersion model for nonzero entries, we have analyzed the effect of (g, h) on the performance, as shown in Figure 2. Therefore, one can obtain satisfying results by following our instruction to set hyperparameters.

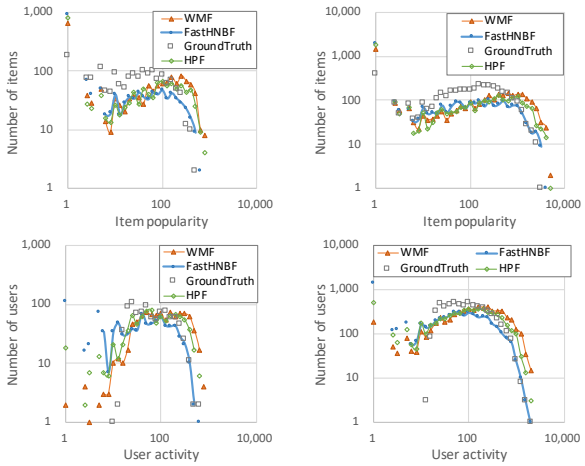
The effect of the long-tail of users and items



(a) Last.fm1K (b) Last.fm2K

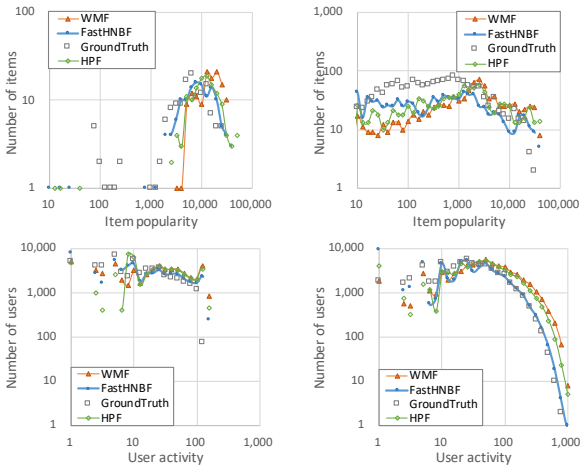
Figure 3: The long-tail of users and items on implicit count data

The effect of the long-tail on implicit count. Figure 3 shows the distributions of user activity and item popularity on implicit count datasets. Since HPF does not perform well on the implicit count datasets due to the data overdispersion, we use BHPF, which truncates the entry values at 1, for the discussion of the long-tail. The predicted values from BHPF larger than 0.1 are set to 1, and the rest are set to 0. WMF, a Gaussian-based model, overestimates both the item pop-



(a) MovieLens100K (b) MovieLens1M

Figure 4: The long-tail of users and items on explicit rating data



(a) Jester2 (b) EachMovie

Figure 5: The long-tail of users and items on "tall-and-skinny" data

ularity and user activity. Notice that both WMF and BHPF are for binarized implicit data, and BHPF can more tightly fit the long-tail of users and items, whereas WMF often overestimates them. However, on a sparse dataset, e.g., Last.fm2K, BHPF underestimates user popularity and item popularity. If the entry value is considered, both entry values and the density of the utility matrix (i.e., how many nonzero entries) affect a model. In this case, a model should be equipped with a component to estimate the data dispersion. On the other hand, when all the nonzero entries are set to 1 (i.e., entry values are truncated at 1), data overdispersion is avoided, and only the density of the utility matrix significantly impacts the predicted values. Hence, BHPF can be adaptive to overdispersed data but be a little sensitive to density. Compared with BHPF, FastHNBF slightly overestimates the user popularity and the item popularity, since FastHNBF directly fit the count value rather than its binarized value truncated at 1. Even though the data are overdispersed, FastHNBF can fit the long-tails well since FastHNBF has a hierarchical dispersion model to estimate the data dispersion.

The effect of the long-tail on explicit rating. Figure 4 shows the distributions of user activity and item popularity on explicit rating datasets. Since the value range of rating data is limited (i.e., in $[0, 10]$), we use HPF rather than BHPF in the experiment. All the predicted values larger than 0.8 are set to 1, and the rest are set to 0. The three methods (i.e., WMF, FastHNBF, and HPF) fit the long-tail well. WMF and HPF overestimate user activity and item popularity slightly, whereas FastHNBF underestimates them. The underestimation of FastHNBF is caused by its dispersion model. According to the likelihood $p(x_{ui}|r_{ui}\boldsymbol{\theta}_u\boldsymbol{\beta}_i)$, when dispersion r_{ui} is larger, inference score $\boldsymbol{\theta}_u\boldsymbol{\beta}_i)$ should be smaller to maximize the likelihood. As a result, the inference score from FastHNBF is underestimated. Accordingly, both the user activity and the item popularity are underestimated as well. On the other hand, compared with the underestimation of BHPF on implicit count datasets, the user activity and the item popularity predicted by HPF are slightly overestimated. We explain that a model with no component for estimating data dispersion, such as HPF, is prone to overestimate the inference score. Recall that only $\boldsymbol{\theta}_u\boldsymbol{\beta}_i$ can be modulated in HPF to maximize the likelihood $p(x_{ui}|\boldsymbol{\theta}_u\boldsymbol{\beta}_i)$ so that the values of nonzero entries would raise the values of latent factors $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$, thus increasing the inference scores of zero entries.

The effect of the long-tail on "tall-and-skinny" utility matrices. We conduct the experiment on two explicit rating datasets: Jester2 and EachMovie. Since the value range of rating data is limited, we use HPF rather than BHPF in the experiment. The result is similar to the experimental result on explicit rating data. FastHNBF fits the user activity and item popularity the best, whereas HPF overestimates slightly. The Gaussian-based model (i.e., WMF) fed by binarized data truncated at 1, let alone the original data without truncation, overestimates the worst. On the other hand, the distributions over item popularity do not follow power laws because the number of the items is much less than the number of the users in the two datasets. Especially in Jester2, which comprises 140 items merely, the distribution over item popularity has a peak centered at 6,300. This observation could inspire future research in the community to improve the recommendation on implicit data via studying which data properties affect the performance of Gaussian-based matrix factorization. Furthermore, studying the difference between Poisson-based models and Gaussian-based ones from the perspective of data properties could help bridge the gap between the two main streams of recommendation models.

Consequently, FastHNBF fits the distributions over user activity and item popularity the best due to its hierarchical dispersion model. With an assist from the dispersion model, the inference model need not fit overdispersed data directly because the dispersion model can absorb the noise in data.

Hence, the three experiments indicate that FastHNBF better represents real data in terms of the ability to capture distributions of user activity and item popularity.

Test of Significance

To show the significance of the performance of Poisson-based models, we do the t-tests. For each pair of models, we first do the f-tests to determine the significance of the equality of variances. When the p-value is higher than 0.05, we then do the t-test assuming equal variances. On the contrary, when the p-value is lower than 0.05, we then do the t-test assuming unequal variances.

Significance of performance on implicit count data

The t-tests of the performance of the comparing methods on implicit count data are shown in Figure 6. The outperformance of FastHNBF on Last.fm1K in terms of precision@5, recall@5, and nDCG@5 is significant. FastHNBF also outperforms other competitors in terms of precision@5 and recall@5 on Last.fm2K. Notice that the p-value of the t-test between the performance of FastHNBF and PRPF on Last.fm2K is 0.096, as shown in Figure 6 (f). The outperformance over PRPF on Last.fm2K in terms of nDCG@5 is insignificant. A similar situation is found on Last.fm360K: the outperformance of FastHNBF over PRPF is significant in all metrics except for nDCG@5. Both BHPF and PRPF are able to mitigate data overdispersion. PRPF considers items permutation for each user, whereas BHPF neglects the values of positive entries by clipping them to 1. Since PRPF considers the ranking over items, PRPF performs better than BHPF in most cases. Except for FastHNBF, there is no Poisson-based method that outperforms the others in all cases. Hence, despite the insignificance, we can conclude that FastHNBF leads to the best and robust performance in all metrics.

Significance of performance on explicit rating data

The t-tests between the performance of the comparing methods on explicit rating data are shown in Figure 7. On MovieLens100K, FastHNBF performs the best, followed by PRPF. PRPF performs slightly better than HPF. The outperformance of PRPF over HPF is insignificant (i.e., the p-values are 0.182, 0.251, and 0.250 in terms of precision@5, recall@5, and nDCG@5, respectively) Compared with this, the outperformance of FastHNBF over PRPF is more significant. The p-value in terms of precision@5 is 0.108, as shown in Figure 7 (a); the p-value in terms of recall@5 is 0.098, as shown in Figure 7 (b). FastHNBF significantly outperforms the other competitors in terms of nDCG@5.

On MovieLens1M, FastHNBF performs the best, followed by NBF and HPF. We can see that NBF and HPF performs similar (i.e., the p-values are 0.786, 0.676, and 0.978 in terms of precision@5, recall@5, and nDCG@5, respectively) Compared with this, the outperformance of FastHNBF over the other methods is more significant, where the maximum p-value is 0.051 (i.e., the outperformance over NBF in terms of nDCG@5).

On MovieLens20M, HPF performs better than BHPF in precision@5 and nDCG@5 but worse than BHPF in terms of recall@5. Compared with the performance gap between HPF and BHPF, FastHNBF reaches higher scores in terms of precision@5 more significantly (with p-value 0.059) We explain that the relevant items are likely ranked in the top 5 by FastHNBF, whereas HPF puts more emphasis on highly ranked items. That is why HPF performs worse than FastHNBF in terms of precision@5 but performs similarly to FastHNBF.

All in all, the Poisson-based methods perform similarly on explicit rating data since the rating is not Poisson distributed. Even so, FastHNBF gains a significant performance improvement, and thus FastHNBF could bridge the gap between Gaussian-based exposure models (e.g., WMF and ExpoMF) and Poisson-based ones.

Significance of performance on data with $M \gg N$

In terms of precision@5, FastHNBF outperforms the other competitors on Jester2 (Figure 8 (d)) and EachMovie (Figure 8 (a)). BHPF reaches a higher recall@5 score than FastHNBF insignificantly (with p-value 0.602). In terms of nDCG@5, on Jester2, FastHNBF performs the best, followed by NBF. Compared with other performance significance, the outperformance of FastHNBF over NBF is more significant since the p-value is the minimum, 0.126, as shown in 8 (c). On EachMovie, FastHNBF slightly outperforms BHPF in terms of recall@5 (with p-value 0.767 as shown in Figure 8 (e)) and nDCG@5 (with p-value 0.478 as shown in Figure 8 (f)). In terms of recall@5, FastHNBF, HPF, and BHPF perform similarly, and they reach recall@5 scores significantly higher than the other three, which also perform similarly, as shown in Figure 8 (e). Both methods perform better than the others significantly in nDCG@5, as shown in Figure 8 (f).

References

Basbug, M. E.; and Engelhardt, B. E. 2017. Coupled Compound Poisson Factorization. *arXiv preprint arXiv:1701.02058* .

Gouvert, O.; Oberlin, T.; and Févotte, C. 2018. Negative Binomial Matrix Factorization for Recommender Systems. *arXiv preprint arXiv:1801.01708* .

Prec@5	HPF	BHPF	PRPF	CCPF+HPF	NBF	FastHNBF
HPF	1.000	0.000	0.000	0.085	0.009	0.000
BHPF	0.000	1.000	0.012	0.000	0.000	0.000
PRPF	0.000	0.012	1.000	0.000	0.000	0.000
CCPF+HPF	0.085	0.000	0.000	1.000	0.006	0.000
NBF	0.009	0.000	0.000	0.006	1.000	0.000
FastHNBF	0.000	0.000	0.000	0.000	0.000	1.000

(a) Precision@5 on Last.fm1K

Recall@5	HPF	BHPF	PRPF	CCPF+HPF	NBF	FastHNBF
HPF	1.000	0.000	0.000	0.563	0.004	0.000
BHPF	0.000	1.000	0.172	0.000	0.000	0.000
PRPF	0.000	0.172	1.000	0.000	0.000	0.000
CCPF+HPF	0.563	0.000	0.000	1.000	0.008	0.000
NBF	0.004	0.000	0.000	0.008	1.000	0.000
FastHNBF	0.000	0.000	0.000	0.000	0.000	1.000

(b) Recall@5 on Last.fm1K

nDCG@5	HPF	BHPF	PRPF	CCPF+HPF	NBF	FastHNBF
HPF	1.000	0.000	0.000	0.120	0.007	0.000
BHPF	0.000	1.000	0.041	0.000	0.000	0.000
PRPF	0.000	0.041	1.000	0.000	0.000	0.000
CCPF+HPF	0.120	0.000	0.000	1.000	0.004	0.000
NBF	0.007	0.000	0.000	0.004	1.000	0.000
FastHNBF	0.000	0.000	0.000	0.000	0.000	1.000

(c) nDCG@5 on Last.fm1K

Prec@5	HPF	BHPF	PRPF	CCPF+HPF	NBF	FastHNBF
HPF	1.000	0.000	0.000	0.042	0.026	0.000
BHPF	0.000	1.000	0.014	0.000	0.000	0.000
PRPF	0.000	0.014	1.000	0.000	0.000	0.024
CCPF+HPF	0.042	0.000	0.000	1.000	0.008	0.000
NBF	0.026	0.000	0.000	0.008	1.000	0.000
FastHNBF	0.000	0.000	0.024	0.000	0.000	1.000

(d) Precision@5 on Last.fm2K

Recall@5	HPF	BHPF	PRPF	CCPF+HPF	NBF	FastHNBF
HPF	1.000	0.000	0.000	0.050	0.011	0.000
BHPF	0.000	1.000	0.004	0.000	0.000	0.000
PRPF	0.000	0.004	1.000	0.000	0.000	0.015
CCPF+HPF	0.050	0.000	0.000	1.000	0.007	0.000
NBF	0.011	0.000	0.000	0.007	1.000	0.000
FastHNBF	0.000	0.000	0.015	0.000	0.000	1.000

(e) Recall@5 on Last.fm2K

nDCG@5	HPF	BHPF	PRPF	CCPF+HPF	NBF	FastHNBF
HPF	1.000	0.000	0.000	0.020	0.040	0.000
BHPF	0.000	1.000	0.032	0.000	0.000	0.000
PRPF	0.000	0.032	1.000	0.000	0.000	0.096
CCPF+HPF	0.020	0.000	0.000	1.000	0.006	0.000
NBF	0.040	0.000	0.000	0.006	1.000	0.000
FastHNBF	0.000	0.000	0.096	0.000	0.000	1.000

(f) nDCG@5 on Last.fm2K

Prec@5	HPF	BHPF	PRPF	FastHNBF
HPF	1.000	0.000	0.000	0.000
BHPF	0.000	1.000	0.133	0.028
PRPF	0.000	0.133	1.000	0.001
FastHNBF	0.000	0.028	0.001	1.000

(g) Precision@5 on Last.fm360K

Recall@5	HPF	BHPF	PRPF	FastHNBF
HPF	1.000	0.000	0.000	0.000
BHPF	0.000	1.000	0.119	0.046
PRPF	0.000	0.119	1.000	0.001
FastHNBF	0.000	0.046	0.001	1.000

(h) Recall@5 on Last.fm360K

nDCG@5	HPF	BHPF	PRPF	FastHNBF
HPF	1.000	0.000	0.000	0.000
BHPF	0.000	1.000	0.061	0.231
PRPF	0.000	0.061	1.000	0.002
FastHNBF	0.000	0.231	0.002	1.000

(i) nDCG@5 on Last.fm360K

Figure 6: T-test on implicit count datasets.

Prec@5	HPF	BHPF	PRPF	CCPF+HPF	NBF	FastHNBF
HPF	1.000	0.125	0.182	0.021	0.550	0.008
BHPF	0.125	1.000	0.006	0.219	0.447	0.000
PRPF	0.182	0.006	1.000	0.001	0.068	0.108
CCPF+HPF	0.021	0.219	0.001	1.000	0.095	0.000
NBF	0.550	0.447	0.068	0.095	1.000	0.003
FastHNBF	0.008	0.000	0.108	0.000	0.003	1.000

(a) Precision@5 on MovieLens100K

Recall@5	HPF	BHPF	PRPF	CCPF+HPF	NBF	FastHNBF
HPF	1.000	0.586	0.251	0.006	0.674	0.017
BHPF	0.586	1.000	0.500	0.001	0.343	0.033
PRPF	0.251	0.500	1.000	0.000	0.135	0.098
CCPF+HPF	0.006	0.001	0.000	1.000	0.025	0.000
NBF	0.674	0.343	0.135	0.025	1.000	0.016
FastHNBF	0.017	0.033	0.098	0.000	0.016	1.000

(b) Recall@5 on MovieLens100K

nDCG@5	HPF	BHPF	PRPF	CCPF+HPF	NBF	FastHNBF
HPF	1.000	0.000	0.250	0.011	0.749	0.037
BHPF	0.000	1.000	0.000	0.126	0.001	0.000
PRPF	0.250	0.000	1.000	0.045	0.505	0.001
CCPF+HPF	0.011	0.126	0.045	1.000	0.033	0.000
NBF	0.749	0.001	0.505	0.033	1.000	0.026
FastHNBF	0.037	0.000	0.001	0.000	0.026	1.000

(c) nDCG@5 on MovieLens100K

Prec@5	HPF	BHPF	PRPF	CCPF+HPF	NBF	FastHNBF
HPF	1.000	0.241	0.030	0.271	0.786	0.004
BHPF	0.241	1.000	0.116	0.809	0.455	0.000
PRPF	0.030	0.116	1.000	0.300	0.087	0.000
CCPF+HPF	0.271	0.809	0.300	1.000	0.434	0.001
NBF	0.786	0.455	0.087	0.434	1.000	0.003
FastHNBF	0.004	0.000	0.000	0.001	0.003	1.000

(d) Precision@5 on MovieLens1M

Recall@5	HPF	BHPF	PRPF	CCPF+HPF	NBF	FastHNBF
HPF	1.000	0.737	0.487	0.898	0.676	0.008
BHPF	0.737	1.000	0.357	0.683	0.912	0.023
PRPF	0.487	0.357	1.000	0.590	0.327	0.004
CCPF+HPF	0.898	0.683	0.590	1.000	0.569	0.007
NBF	0.676	0.912	0.327	0.569	1.000	0.029
FastHNBF	0.008	0.023	0.004	0.007	0.029	1.000

(e) Recall@5 on MovieLens1M

nDCG@5	HPF	BHPF	PRPF	CCPF+HPF	NBF	FastHNBF
HPF	1.000	0.000	0.000	0.159	0.978	0.021
BHPF	0.000	1.000	0.165	0.000	0.000	0.000
PRPF	0.000	0.165	1.000	0.043	0.005	0.000
CCPF+HPF	0.159	0.000	0.043	1.000	0.256	0.002
NBF	0.978	0.000	0.005	0.256	1.000	0.051
FastHNBF	0.021	0.000	0.000	0.002	0.051	1.000

(f) nDCG@5 on MovieLens1M

Prec@5	HPF	BHPF	FastHNBF
HPF	1.000	0.278	0.059
BHPF	0.278	1.000	0.006
FastHNBF	0.059	0.006	1.000

(g) Precision@5 on MovieLens20M

Recall@5	HPF	BHPF	FastHNBF
HPF	1.000	0.094	0.019
BHPF	0.094	1.000	0.745
FastHNBF	0.019	0.745	1.000

(h) Recall@5 on MovieLens20M

nDCG@5	HPF	BHPF	FastHNBF
HPF	1.000	0.000	0.996
BHPF	0.000	1.000	0.000
FastHNBF	0.996	0.000	1.000

(i) nDCG@5 on MovieLens20M

Figure 7: T-test on explicit rating datasets.

Prec@5	HPF	BHPF	PRPF	CCPF+HPF	NBF	FastHNBF
HPF	1.000	0.089	0.844	0.497	0.849	0.000
BHPF	0.089	1.000	0.199	0.060	0.190	0.041
PRPF	0.844	0.199	1.000	0.424	0.991	0.002
CCPF+HPF	0.497	0.060	0.424	1.000	0.415	0.001
NBF	0.849	0.190	0.991	0.415	1.000	0.001
FastHNBF	0.000	0.041	0.002	0.001	0.001	1.000

(a) Precision@5 on Jester2

Recall@5	HPF	BHPF	PRPF	CCPF+HPF	NBF	FastHNBF
HPF	1.000	0.000	0.203	0.891	0.819	0.000
BHPF	0.000	1.000	0.010	0.005	0.002	0.602
PRPF	0.203	0.010	1.000	0.504	0.452	0.051
CCPF+HPF	0.891	0.005	0.504	1.000	0.966	0.021
NBF	0.819	0.002	0.452	0.966	1.000	0.012
FastHNBF	0.000	0.602	0.051	0.021	0.012	1.000

(b) Recall@5 on Jester2

nDCG@5	HPF	BHPF	PRPF	CCPF+HPF	NBF	FastHNBF
HPF	1.000	0.536	0.200	0.807	0.463	0.003
BHPF	0.536	1.000	0.361	0.809	0.237	0.001
PRPF	0.200	0.361	1.000	0.392	0.172	0.002
CCPF+HPF	0.807	0.809	0.392	1.000	0.465	0.013
NBF	0.463	0.237	0.172	0.465	1.000	0.126
FastHNBF	0.003	0.001	0.002	0.013	0.126	1.000

(c) nDCG@5 on Jester2

Prec@5	HPF	BHPF	PRPF	CCPF+HPF	NBF	FastHNBF
HPF	1.000	0.359	0.462	0.030	0.615	0.001
BHPF	0.359	1.000	0.116	0.003	0.149	0.018
PRPF	0.462	0.116	1.000	0.135	0.806	0.001
CCPF+HPF	0.030	0.003	0.135	1.000	0.070	0.000
NBF	0.615	0.149	0.806	0.070	1.000	0.002
FastHNBF	0.001	0.018	0.001	0.000	0.002	1.000

(d) Precision@5 on EachMovie

Recall@5	HPF	BHPF	PRPF	CCPF+HPF	NBF	FastHNBF
HPF	1.000	0.159	0.159	0.024	0.157	0.171
BHPF	0.159	1.000	0.019	0.001	0.018	0.767
PRPF	0.159	0.019	1.000	0.437	0.998	0.065
CCPF+HPF	0.024	0.001	0.437	1.000	0.436	0.020
NBF	0.157	0.018	0.998	0.436	1.000	0.064
FastHNBF	0.171	0.767	0.065	0.020	0.064	1.000

(e) Recall@5 on EachMovie

nDCG@5	HPF	BHPF	PRPF	CCPF+HPF	NBF	FastHNBF
HPF	1.000	0.018	0.076	0.004	0.397	0.002
BHPF	0.018	1.000	0.000	0.000	0.002	0.478
PRPF	0.076	0.000	1.000	0.504	0.288	0.000
CCPF+HPF	0.004	0.000	0.504	1.000	0.053	0.000
NBF	0.397	0.002	0.288	0.053	1.000	0.001
FastHNBF	0.002	0.478	0.000	0.000	0.001	1.000

(f) nDCG@5 on EachMovie

Figure 8: T-test on explicit rating datasets with $M \gg N$.