Nonparametric Poisson Factorization Machine

Avijit Saha†, Ayan Acharya*, Balaraman Ravindran†, Joydeep Ghosh*
†{avijit,ravi}@cse.iitm.ac.in, *{aacharya,jghosh}@utexas.edu,
†Department of CS, Indian Institute of Technology, Chennai, *Department of ECE, University of Texas at Austin

***Abstract*—Factorization Machine (FM) provides a generic framework that combines the prediction quality of factorization models with the flexibility of feature engineering that discriminative models like SVM offer. The Bayesian Factorization Machine [10], with its impressive predictive performance and the convenience of automatic tuning of parameters, has been one of the most successful and efficient approaches within this framework. However, this model has two major drawbacks. Firstly, it assumes that the data is generated from Gaussian distributions that may not be the best assumption for count data such as integer-valued ratings. Secondly, to get the best performance, one needs to cross-validate over the number of latent factors used for modeling the pairwise interaction in FM, a process that is computationally intensive. This paper introduces the Nonparametric Poisson Factorization Machine (NPFM), which models count data using the Poisson distribution, which provides both modeling and computational advantages for sparse data. The ideal number of latent factors is estimated from the data itself, thereby addressing a key limitation of existing approaches to FM. Additionally, NPFM has linear time complexity with respect to the number of non-zero observations.**

Keywords*—*factorization machine, matrix factorization, tensor factorization, gamma process, recommender systems.

I. INTRODUCTION

Factorization models have received extensive attention in the data mining community recently for certain problems characterized by high-dimensional, sparse matrices, due to their simplicity, prediction quality and scalability. One of the most successful domains for factorization models have been recommender systems. Perhaps the most well studied factorization model is matrix factorization (MF) [20], [21], [27], [28], [11], [14] using the Frobenius norm as the loss function. Several specialized factorization models have been proposed specific to particular problems, for instance: SVD++ [18], TimeSVD++ [19], Factorizing Personalized Markov Chains for Next-Basket Recommendation (FPMC) [25], and Bayesian temporal collaborative filtering with Bayesian probabilistic tensor factorization (BPTF) [34]. Also many learning and inference methods have been studied for factorization models, for example: stochastic gradient descent [20], alternating least-squares [39], variational Bayes [22], [17], [30], and Markov chain Monto Carlo (MCMC) Gibbs sampling inference [28].

In more challenging prediction scenarios where additional "side-information" is available and/or higher order features may be needed, new challenges due to feature engineering needs arise. While interaction terms are typically desired in such scenarios, the number of such terms grows very quickly. This dilemma is cleverly addressed by the Factorization Machine (FM) [24], which combines high prediction quality of factorization models with the flexibility of feature engineering. FM represents data as real-valued features like standard machine learning approaches, such as SVMs, and uses interactions between each pair of variables as well but constrained to a low-dimensional latent space. By restricting the latent space, the number of parameters needed to be determined is kept manageable. Interestingly, the framework of FM subsumes many successful factorization models like matrix factorization [20], SVD++ [18], TimeSVD++ [19], PITF [26], and FPMC [25]. Other advantages of FM include – 1) FM allows parameter estimation with extremely sparse data where SVMs fail; 2) FM has linear complexity, can be optimized in the primal and, unlike SVMs, does not rely on support vectors; and 3) FM is a general predictor that can work with any real valued feature vector, while several state-of-the-art factorization models work only on very restricted input data.

FM is usually learned using stochastic gradient descent (SGD) [20]. An FM that uses SGD for learning is conveniently addressed as SGD-FM in this paper. SGD obviates the need to store the entire dataset in memory and hence is often preferred for large scale learning due to memory and speed considerations [30]. Though SGD is scalable and enjoys local convergence guarantees [29], it requires manual tuning of learning rate and regularization parameters, and may overfit the data. Alternative methods to solve FM include Bayesian Factorization Machine [10] which provides state-of-the-art performance using MCMC sampling as the inference mechanism and avoids expensive manual tuning of the learning rate and regularization parameters (this framework is addressed as MCMC-FM in this paper). However, MCMC-FM assumes that the observations are generated from a Gaussian distribution which, obviously, is not a good fit for count data. Additionally, for both SGD-FM and MCMC-FM, one needs to solve an expensive model selection problem to identify the optimal number of latent factors. We note that alternative models for count data have recently emerged that use discrete distributions provide better interpretability and scale only with the number of non-zero elements [12], [37], [38], [1].

This paper proposes a **N**onparametric **P**oisson **F**actorization **M**achine (NPFM) to overcome the limitations of the existing inference techniques for FM mentioned above. The specific advantages of NPFM include:

- NPFM provides a more interpretable model with a better fit for count datasets.
- NPFM is a nonparametric model and avoids the costly model selection procedure by automatically finding the number of latent factors suitable for modeling the pairwise interaction matrix in FM.
- NPFM takes advantages of the Poisson distribution [12], [1], [3], [2] and considers only sampling over the non-zero entries. On the other hand, existing FM methods, which assume

a Gaussian distribution, must iterate over both positive and negative samples in the implicit setting. Such strategy is expensive for large datasets and often needs to be solved using a costly positive and negative data sampling approach instead. NPFM can take advantages of natural sparsity of the data which existing inference technique of FM fails to exploit.

II. Background and Related Work

This section presents the related literature and the background materials that are useful for understanding the framework described in Section III. We also enlist a few lemmata that are used in the derivation of the Gibbs sampling updates. The derivations of these Lemmas can be found in [1], [3].

A. Negative Binomial Distribution

The negative binomial (NB) distribution $m \sim \text{NB}(r,p)$, with probability mass function (PMF) $P(M=m) = \frac{\Gamma(m+r)}{m!\Gamma(r)}p^m(1-p)^r$ for $m \in \mathbb{Z}$, can be augmented into a gamma-Poisson construction as $m \sim \text{Pois}(\lambda)$, $\lambda \sim \text{Gam}(r, p/(1-p))$, where the gamma distribution is parameterized by its shape r and scale $p/(1-p)$. It can also be augmented under a compound Poisson representation as $m = \sum_{t=1}^{l} u_t, u_t \overset{iid}{\sim} \text{Log}(p), l \sim \text{Pois}(-r\ln(1-p))$, where $u \sim \text{Log}(p)$ is the logarithmic distribution [16]. Consequently, we have the following Lemma.

Lemma 1 ([36]). *If $m \sim \text{NB}(r,p)$ is represented under its compound Poisson representation, then the conditional posterior of l given m and r has PMF:*

$$P(l=j|m,r) = \frac{\Gamma(r)}{\Gamma(m+r)}|s(m,j)|r^j,\ j=0,1,\cdots,m, \quad (1)$$

where $|s(m,j)|$ are unsigned Stirling numbers of the first kind. We denote this conditional posterior as $(l|m,r) \sim \text{CRT}(m,r)$, a Chinese restaurant table (CRT) count random variable, which can be generated via $l = \sum_{n=1}^{m} z_n, z_n \sim \text{Bernoulli}(r/(n-1+r))$.

Lemma 2. *Let $X = \sum_{k=1}^{K} x_k$, $x_k \sim \text{Pois}(\zeta_k)\ \forall k$, and $\zeta = \sum_{k=1}^{K}\zeta_k$. If $(y_1,\cdots,y_K|X) \sim \text{Mult}(X, \zeta_1/\zeta,\cdots,\zeta_K/\zeta)$ and $X \sim \text{Pois}(\zeta)$, then the following holds:*

$$P(X,x_1,\cdots,x_K) = P(X,y_1,\cdots,y_K). \quad (2)$$

Lemma 3. *If $x_i \sim \text{Pois}(m_i\lambda)$, $\lambda \sim \text{Gam}(r,1/c)$, then $x = \sum_i x_i \sim \text{NB}(r,p)$, where $p = (\sum_i m_i)/(c+\sum_i m_i)$.*

Lemma 4. *If $x_i \sim \text{Pois}(m_i\lambda)$, $\lambda \sim \text{Gam}(r,1/c)$, then*

$$(\lambda|\{x_i\},r,c) \sim \text{Gam}\left(r+\sum_i x_i, \frac{1}{c+\sum_i m_i}\right). \quad (3)$$

Lemma 5. *If $r_i \sim \text{Gam}(a_i,1/b)\ \forall i$, $b \sim \text{Gam}(c,1/d)$, then we have:*

$$(b|\{r_i,a_i\},c,d) \sim \text{Gam}\left(\sum_i a_i + c, \frac{1}{\sum_i r_i + d}\right). \quad (4)$$

Lemma 6. *If $x_i \sim \text{Pois}(m_i r_2)$, $r_2 \sim \text{Gam}(r_1,1/d)$, $r_1 \sim \text{Gam}(a,1/b)$, , then $(r_1|-) \sim \text{Gam}(a+\ell, 1/(b-log(1-p)))$ where $(\ell|x,r_1) \sim \text{CRT}(\sum_i x_i, r_1), p = \sum_i m_i/(d+\sum_i m_i)$.*

B. Gamma Process

The gamma process [9], [32], [33], [32], [1], [3] $G \sim \Gamma\text{P}(c,H)$ is a completely random measure defined on the product space $\mathbb{R}_+ \times \Omega$, with concentration parameter c and a finite and continuous base measure H over a complete separable metric space Ω, such that $G(A_i) \sim \text{Gam}(H(A_i),1/c)$ are independent gamma random variables for disjoint partition $\{A_i\}_i$ of Ω. The Lévy measure of the gamma process can be expressed as $\nu(drd\omega) = r^{-1}e^{-cr}drH(d\omega)$. Since the Poisson intensity $\nu^+ = \nu(\mathbb{R}_+\times\Omega) = \infty$ and the value of $\int_{\mathbb{R}_+\times\Omega} r\nu(drd\omega)$ is finite, a draw from the gamma process consists of countably infinite atoms, which can be expressed as follows:

$$G = \sum_{k=1}^{\infty} r_k\delta_{\omega_k},\ (r_k,\omega_k) \overset{iid}{\sim} \pi(drd\omega),\ \pi(drd\omega)\nu^+ \equiv \nu(drd\omega). \quad (5)$$

C. Factorization Machine (FM)

FM combines the advantages of feature based methods like SVMs with factorization models. Popular feature based methods like SVMs can be learnt using standard tools like LIBSVM [8] and SVM-Light [15]. But feature based methods encounter problems when facing high-dimensional but sparse dyadic data where factorization models have been more successful. An FM learns a function $f : \mathbb{R}^D \to T$ which is a mapping from a real valued feature vector $\boldsymbol{X} \in \mathbb{R}^D$ to a target domain T. The training data for FM consists of N tuples, each of which contains the covariates $\boldsymbol{x}_n$ and the response variable y_n. The underlying model equation for FM is:

$$y = w_0 + \sum_{i=1}^{D} x_i w_i + \sum_{i=1}^{D}\sum_{j=i+1}^{D} x_i x_j \langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle, \quad (6)$$

where w_0 is the global bias, w_i is the bias associated with i^{th} variable, and $\boldsymbol{v}_i$ is the latent factor vector of dimension K associated with the i^{th} variable. $\langle \boldsymbol{v}_i, \boldsymbol{v}_j\rangle$ models the interaction between the i^{th} and j^{th} feature. The objective is to estimate the parameters $w_0 \in \mathbb{R}$, $\boldsymbol{w} \in \mathbb{R}^D$, and $\boldsymbol{V} \in \mathbb{R}^{D\times K}$. Instead of using a parameter $w_{i,j} \in \mathbb{R}$ for each interaction, FM models the pairwise interaction by factorizing it. Since for any positive definite matrix $\boldsymbol{W}$, there exists a matrix $\boldsymbol{V}$ such that $W = \boldsymbol{V}\boldsymbol{V}^\dagger$ provided K is sufficiently large, FM can express any interaction matrix $\boldsymbol{W}$. This is a remarkably smart way to express pairwise interaction in big sparse datasets. In fact, many of the existing collaborative filtering algorithms aim to do the same, but with the specific goal of user-item recommendation and thus fail to recognize the underlying mathematical basis that FM successfully discovers.

D. Poisson Factor Analysis

Since the pairwise interaction matrix in NPFM is modeled using a Poisson factorization, a brief discussion on Poisson factor analysis is necessary. A large number of discrete latent variable models for count matrix factorization can be united under Poisson factor analysis (PFA) [38], [37], [1], [3], [2], which factorizes a count matrix $\mathbf{Y} \in \mathbb{Z}^{D\times V}$ under the Poisson likelihood as $\mathbf{Y} \sim \text{Pois}(\boldsymbol{\Phi}\boldsymbol{\Theta})$, where $\boldsymbol{\Phi} \in \mathbb{R}_+^{D\times K}$ is the factor loading matrix or dictionary, $\boldsymbol{\Theta} \in \mathbb{R}_+^{K\times V}$ is the factor score matrix. A wide variety of algorithms, although constructed with different motivations and for distinct problems, can all be

viewed as PFA with different prior distributions imposed on $\mathbf{\Phi}$ and $\mathbf{\Theta}$. For example, non-negative matrix factorization [7], with the objective to minimize the Kullback-Leibler divergence between $\mathbf{N}$ and its factorization $\mathbf{\Phi\Theta}$, is essentially PFA solved with maximum likelihood estimation. LDA [6] is equivalent to PFA, in terms of both block Gibbs sampling and variational inference [11], [12], if Dirichlet distribution priors are imposed on both $\boldsymbol{\phi}_k \in \mathbb{R}_+^D$, the columns of $\mathbf{\Phi}$, and $\boldsymbol{\theta}_k \in \mathbb{R}_+^V$, the columns of $\mathbf{\Theta}$. Interested readers are referred to [38], [37], [1], [3], [2] for further examples of models that utilize PFA.

III. NONPARAMETRIC POISSON FACTORIZATION MACHINE

A. MODEL

Consider a training data consisting of N tuples of the form $(\boldsymbol{x}_n, y_n)_{n=1}^N$ where $\boldsymbol{x}_n$ is the feature representation and y_n is the associated response variable for the n^{th} training instance. In NPFM, the response variable $y \in \mathbb{Z}$ is assumed to be linked to the covariate $\boldsymbol{x} \in \mathbb{R}_+^D$ as:

$$y \sim \text{Pois}\left(w_0 + \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \sum_{(i,j):i<j} x_i x_j \sum_{k=1}^{\infty} r_k v_{ik} v_{jk}\right). \quad (7)$$

According to the standard terminology in the literature of recommender systems, w_0 denotes the global bias and is sampled as $w_0 \sim \text{Gam}(\alpha_0, 1/\beta_0)$. $\boldsymbol{w} = (w_i)_{i=1}^D \in \mathbb{R}_+^D$ and w_i indicates the strength of the i^{th} variable and can be thought of as the "bias" corresponding to the i^{th} feature. w_i is modeled as $w_i \sim \text{Gam}(\alpha_i, 1/\beta_i)\ \forall i \in \{1, 2, \cdots, D\}$. A gamma process $G \sim \Gamma\text{P}(G_0, c)$ is further introduced, a draw from which is expressed as $G = \sum_{k=1}^{\infty} r_k \delta_{\boldsymbol{v}_k}$, where $\boldsymbol{v}_k = (v_{ik})_{i=1}^D$ is an atom drawn from a D-dimensional base distribution as $\boldsymbol{v}_k \sim \prod_i \text{Gam}(\zeta_i, 1/\delta_k)$ and $r_k = G(\boldsymbol{v}_k)$ is the associated weight. The i^{th} variable is associated with the infinite dimensional latent factor vector $\boldsymbol{v_i}$. The objective is to learn a distribution over the weights w_0, $\boldsymbol{w}$, $\{r_k, \boldsymbol{v}_k\}$ based on the training observations. To complete the generative process, gamma priors are imposed on the parameters as:

$$\alpha_0 \sim \text{Gam}(a_0, 1/b_0), \beta_0 \sim \text{Gam}(c_0, 1/d_0), \delta_k \sim \text{Gam}(g_k, 1/h_k).$$

$$\alpha_i \sim \text{Gam}(a_i, 1/b_i), \beta_i \sim \text{Gam}(c_i, 1/d_i), \zeta_i \sim \text{Gam}(e_i, 1/f_i).$$

The pairwise interaction matrix $\boldsymbol{W}$ is modeled little differently in NPFM compared to other existing formulations of FM. In NPFM, $\boldsymbol{W}$ is factored as $w_{ij} \sim \text{Pois}\left(\sum_k r_k v_{ik} v_{jk}\right)$, where r_k denotes the strength of the k^{th} latent factor and v_{ik} denotes the affinity of the i^{th} variable towards the k^{th} latent factor. Note that such assumptions have been successfully used already for network analysis [4], [35] and count data modeling [1] and is similar to using eigen decomposition of the matrix $\boldsymbol{W}$ with integer entries. However, unlike in eigen decomposition, the factors v_{ik}'s are neither normalized nor do they form an orthogonal set of basis vectors. Of course, by sampling the entries of $\boldsymbol{W}$ from a Poisson distribution, we do restrict these entries to integers, which is yet another departure from the standard formulation of FM. However, the empirical results reveal that this is not at all an unreasonable assumption. The gamma process $G = \sum_{k=1}^{\infty} r_k \delta_{\boldsymbol{v}_k}$ allows to estimate the ideal number of latent factors from the data itself, without any need to cross-validate the performance with varying number of latent factors. The factor specific variable r_k adds more flexibility to the model. For example, if there is a need for modeling temporal count datasets, such as in a recommender system that evolves over time, r_k's can be linked across successive time stamps using a gamma Markov chain [1], [2]. This would imply that a certain combination of user-item pair changes its characteristics over time. Since in practical recommender systems, such evolution is very natural, we prefer to maintain these latent variables. In Section III-C, we illustrate PPFM, a simplified version of NPFM, which does not use the variables r_k's at all. In Section IV, we see that the performances of NPFM and PPFM are comparable, implying that the added flexibility does not hurt the predictive performance.

B. INFERENCE USING GIBBS SAMPLING

Though NPFM supports countably infinite number of latent factors, in practice, it is impossible to instantiate all of them. Instead of marginalizing out the underlying stochastic process [5], [23] or using slice sampling [31] for nonparametric modeling, for simplicity, a finite approximation of the infinite model is considered by truncating the number of latent factors K, by letting $r_k \sim \text{Gam}(\gamma_0/K, 1/c)$. Such an approximation approaches the original infinite model as K approaches infinity. Further, gamma priors are imposed on both γ_0 and c as $\gamma_0 \sim \text{Gam}(e_0, 1/f_0),\ c \sim \text{Gam}(g_0, 1/h_0)$.

For each $(\boldsymbol{x}_n, y_n)$, consider a vector of latent count variables $\boldsymbol{z}_n$, which is assumed to consist of three parts:

$$\boldsymbol{z}_n = \left(z_{1n}, (z_{2ni})_{i=1}^D, (z_{3nijk})_{(i,j):i<j;k}\right),$$

where $z_{1n} \sim \text{Pois}(w_0)$, $z_{2ni} \sim \text{Pois}(x_{ni} w_i)$, and $z_{3nijk} \sim \text{Pois}(x_{ni} x_{nj} r_k v_{ik} v_{jk})$. As a sum of Poisson random variables is itself a Poisson with rate equal to the sum of the rates, one gets the following:

$$y_n = z_{1n} + \sum_{i=1}^{D} z_{2ni} + \sum_{(i,j):i<j;k} z_{3nijk}.$$

Note that when $y_n = 0$, $\boldsymbol{z}_n = 0$ with probability 1 and hence, NPFM needs to sample $\boldsymbol{z}_n$ only for the non-zero entries while doing inference. Using Lemma 2, the conditional posterior of these latent counts can be expressed as follows:

$$\boldsymbol{z}_n | \sim \text{Mult}\left(\frac{w_0, (w_i x_{ni})_{i=1}^D, (x_{ni} x_{nj} r_k v_{ik} v_{jk})_{(i,j):i<j;k}}{w_0 + \sum_i w_i x_{ni} + \sum_{(i,j):i<j} x_{ni} x_{nj} \sum_{k=1}^{K} r_k v_{ik} v_{jk}}; y_n\right). \quad (8)$$

Sampling of w_0 : Using Lemma 4, the conditional posterior of w_0 can be expressed as:

$$w_0 | \sim \text{Gam}\left(\alpha_0 + z_{1.}, 1/(\beta_0 + N)\right). \quad (9)$$

Sampling of w_i : Using Lemma 4, the conditional posterior of w_i can be expressed as:

$$w_i | \sim \text{Gam}\left(\alpha_i + z_{2.i}, 1/(\beta_i + x_{.i})\right). \quad (10)$$

Sampling of v_{ik} : Using Lemma 4, the conditional posterior of v_{ik} can be expressed as:

$$v_{ik}| \sim \text{Gam}\left(\zeta_i + z_{3.ik}, 1/(\delta_k + \sum_{n=1}^{N} r_k x_{ni} \sum_{j \neq i} x_{nj} v_{jk}\right). \quad (11)$$

Sampling of r_k : Using Lemma 4, the conditional posterior of r_k can be expressed as:

$$r_k| \sim \text{Gam}\left(\gamma_0/K + q_k, 1/(c + s_k)\right), \quad (12)$$

$$q_k = \sum_{n=1}^{N} \sum_{(i,j):i<j} z_{3nijk}, s_k = \sum_{n=1}^{N} \sum_{(i,j):i<j} x_{ni} x_{nj} v_{ik} v_{jk}.$$

Sampling of β_0 : Using Lemma 5, the conditional posterior of β_0 can be expressed as:

$$\beta_0| \sim \text{Gam}(c_0 + \alpha_0, 1/(d_0 + w_0). \quad (13)$$

Sampling of β_i : Using Lemma 5, the conditional posterior of β_i can be expressed as:

$$\beta_i| \sim \text{Gam}(c_i + \alpha_i, 1/(d_i + w_i)). \quad (14)$$

Sampling of δ_k : Using Lemma 5, the conditional posterior of δ_k can be expressed as:

$$\delta_k| \sim \text{Gam}(g_k + \zeta_., 1/(h_k + v_{.k})). \quad (15)$$

Sampling of c : Using Lemma 5, the conditional posterior of c can be expressed as:

$$c| \sim \text{Gam}(\gamma_0 + g_0, 1/(h_0 + r_.)). \quad (16)$$

Sampling of α_0 : Straightforward sampling of α_0 is not possible as this is the shape parameter of a gamma distribution and the prior and the likelihood are not conjugate. However, from the generative assumptions, we have $z_{1.} \sim \text{Pois}(Nw_0)$. Using Lemma 3, one can have $w_0 \sim \text{NB}(\alpha_0, N/(N + \beta_0))$. Now augment l_0 as $l_0 \sim \text{CRT}\left(z_{1.}, \alpha_0\right)$, and using Lemma 6 one can sample:

$$\alpha_0| \sim \text{Gam}(a_0 + l_0, 1/(b_0 - \log(\beta_0/(\beta_0 + N)))). \quad (17)$$

Sampling of α_i : Straightforward sampling of α_i is not possible as this is the shape parameter of a gamma distribution and the prior and the likelihood are not conjugate. However, from the generative assumptions, we have $z_{2.i} \sim \text{Pois}(w_i m_i)$, where $m_i = \sum_n x_{ni}$. Using Lemma 3, one can have $w_i \sim \text{NB}(\alpha_i, m_i/(m_i + \beta_i))$. Now augment l_i as $l_i \sim \text{CRT}\left(z_{2.i}, \alpha_i\right)$, and using Lemma 6 one can sample:

$$\alpha_i| \sim \text{Gam}(a_i + l_i, 1/(b_i - \log(1 - p_i))). \quad (18)$$

where $p_i = m_i/(\beta_i + m_i)$.

Sampling of ζ_i : Since $z_{3.ik} \sim \text{Pois}(m_{ik} v_{ik})$ where $m_{ik} = \sum_n x_{ni} r_k \sum_{j \neq i} x_{nj} v_{jk}$ and $v_{ik} \sim \text{Gam}(\zeta_i, 1/\delta_k)$, integrating out v_{ik} and using Lemma 3, one has $z_{3.ik} \sim \text{NB}\left(\zeta_i, p_{ik}\right)$ $\forall k \in \{1, 2, \cdots, K\}$ where $p_{ik} = \frac{m_{ik}}{\delta_k + m_{ik}}$. Augment $\ell_{ik} \sim \text{CRT}\left(z_{3.ik}, \zeta_i\right)$ and using Lemma 6 sample ζ_i as follows:

$$\zeta_i| \sim \text{Gam}(e_i + \sum_k \ell_{ik}, 1/(f_i - \sum_k \log(1 - p_{ik}))). \quad (19)$$

Sampling of γ_0 : Since $z_{3...k} \sim \text{Pois}(r_k m_k)$ where $m_k = \sum_{n,\{(i,j):i<j\}} x_{ni} x_{nj} v_{ik} v_{jk}$ and $r_k \sim \text{Gam}(\gamma_0/K, 1/c)$, integrating out r_k and using Lemma 3, one has $z_{3..k} \sim \text{NB}\left(\gamma_0/K, p_k\right)$ where $p_k = m_k/(c + m_k)$. Augment $\ell_k \sim \text{CRT}\left(z_{3...k}, \gamma_0/K\right)$ and using Lemma 6 sample,

$$\gamma_0| \sim \text{Gam}(e_0 + \sum_k \ell_k, 1/(f_0 - 1/K \sum_k \log(1 - p_k))), \quad (20)$$

C. PARAMETRIC VERSION

We also consider a special case of NPFM, **P**arametric **P**oisson **F**actorization **M**achine (PPFM), as a baseline to compare against. The key difference between NPFM and PPFM is that in NPFM, one does not need to tune over the number of latent factors. Even with a finite approximation of NPFM, it is sufficient to set K at a high value and the inference itself predicts the ideal number of latent factors from the data. On the other hand, in PPFM, one needs to choose the number of latent factors by a cross-validation process which is time-consuming and computationally intensive. Additionally, to make the baseline comparable against other existing formulations of FM (see Eq. 6), the term r_k is left out. To be more precise, in PPPF, we consider the following generative process for the label y:

$$y \sim \text{Pois}\left(w_0 + \langle \boldsymbol{w}, \boldsymbol{x} \rangle + \sum_{i,j:i<j} x_i x_j \sum_{k=1}^{K} v_{ik} v_{jk}\right). \quad (21)$$

Here, $w_0 \sim \text{Gam}(\alpha_0, 1/\beta_0)$, $\boldsymbol{w} = (w_i)_{i=1}^{D} \in \mathbb{R}_+^D$ and $w_i \sim \text{Gam}(\alpha_i, 1/\beta_i)$ $\forall i \in \{1, 2, \cdots, D\}$. Also $v_{ik} \sim \text{Gam}(\zeta_i, 1/\delta_k)$. Similar to NPFM, gamma priors are imposed on the other parameters.

D. TIME AND SPACE COMPLEXITY

NPFM has linear time complexity with respect to the number of non-zero training instances. If S is the number of non-zero training instances in the data, maximum number of latent factors used be K, then NPFM has time complexity of $\text{O}(SKD)$. Direct implementation NPFM is infeasible because one needs to store all the latent count variables for all the observations in training set which leads to a space complexity of $\text{O}(D^2SK)$, which is clearly prohibitive for large datasets like Netflix with 100 million observations. To avoid such space complexity, the implementation does not store the latent count variables at all, but instead stores the summary statistics required in the updates for the Gibbs sampler, such as $z_{1.}$, $z_{2.i}$, and $z_{3.ik}$. This representation helps maintain space complexity of $\text{O}(DK)$.

IV. EXPERIMENTAL RESULTS

We validate both the performance and the runtime of our model on four different movie rating datasets Movielens 100k,1m,10m[1], and Netflix[2], popularly used in the recommender system literature. We evaluate our method on both accuracy and time. To evaluate accuracy, we consider the recommendation problem as a ranking problem and use mean average precision (MAP), mean average recall (MAR), and F-measure as the metrics [11], [12], [13]. We also use mean Normalized Discounted Cumulative Gain (MNDCG) as the performance measure to capture the positional importance of retrieved items. We use time per iteration for measuring the

[1] http://grouplens.org/datasets/movielens/

[2] http://www.netflixprize.com/

time of execution. For all the datasets, we measure the accuracy of prediction on a held out set consisting of 20% data instances randomly selected. Training is done on the remaining 80% of the data instances. During the training phase, the held out set is considered as missing data.

For all the datasets, utmost 10000 users were selected at random [12]. For Movielens 100k and 1M, which had fewer than 10000 users, we used all the users. Let's denote this randomly selected user set by U. Let $Relevant_u$ and $Retrieved_u$ be the set of relevant and retrieved items for user u. Further, let M be the set of all movies, T_u be the set of movies present in the training set and rated by user u and P_u be the set of movies present in the held out set and rated by user u. Note that, $Relevant_u = P_u$. The set of unconsumed items for user u is given by $M - T_u$. For each user u in U, we predict the rating score for all her unconsumed items. Then for each user we select top 100 items according to the predicted rating. These top 100 items are considered as the $Retrieved$ set for that user. Using these definitions of $Relevant_u$ and $Retrieved_u$, we calculate the $\text{Precision}_u@P$ and $\text{Recall}_u@P$ for each user u in U by standard Information Retrieval definition. We calculate MAP and MAR by averaging the precision and recall score from all the users as follows:

$$MAP = \sum_{u \in U} Precision_u@P/|U|, MAR = \sum_{u \in U} Recall_u@P/|U|. \quad (22)$$

The F-measure is the harmonic mean of MAP and MAR. We calculate Discounted Cumulative Gain at position P ($\text{DCG}_u@P$) for each user u in U as follows:

$$DCG_u@P = rel_u(1) + \sum_{k=2}^{P} rel_u(k)/\log_2(k), \quad (23)$$

where, $rel_u(i)$ is 1 if the item is present in $Relevant_u$, otherwise 0. We consider the $Relevant_u$, as the ideal ordering for user u and calculate $IDCG_u$ based on that. Then we calculate mean NDCG@P (MNDCG@P) as:

$$MNDCG@P = \frac{1}{|U|} \sum_{u \in U} DCG_u@P/IDCG_u@P. \quad (24)$$

A. EXPERIMENTAL SETUP AND PARAMETER SELECTION

We compare NPFM with two strong baselines: SGD-FM [24] and MCMC-FM [10]. Note that in both the SGD-FM and MCMC-FM, Gaussian distributional assumption is made to represent the data likelihood. NPFM selects all the model parameters automatically, but we set the maximum number of factors as 50 for all the experiments. We chose the same latent factor dimension for all the experiments with PPFM, SGD-FM, and MCMC-FM, as well. NPFM, PPFM, and MCMC-FM are based on Gibbs sampling and they need an initial burn-in phase. We used 1500 burn-in iterations and 1000 collection iterations except in the Netflix dataset where 100 burn-in and 100 collection iterations were used due to time constraints. For SGD-FM we found that 300 iterations were sufficient for convergence. So, we ran SGD-FM for 300 iterations for all the datasets except the Netflix dataset where only 200 iterations were considered. The implementation is available at https://github.com/aacharya/NPFM.

In NPFM, all the hyperprior parameters $(a_0, b_0, c_0, d_0, \{a_i, b_i, e_i, f_i\}, h_0, \gamma_0)$ are initialized to 1. We initialize w, v, and r to a small positive value and all other parameters to 0. Likewise, for PPFM, all hyperprior parameters are initialized to 1, w and v are initialized to a small positive value and all other parameters are initialized to 0. We select the learning rate and regularization parameters of SGD-FM using cross validation. We found that SGD-FM performs best with 0.001 learning rate and 0.01 regularization. So we stick to this value for all the experiments of SGD-FM. For SGD-FM, we initialize w_0, w, and v using a Gaussian distribution with 0 mean and 0.01 variance. For MCMC-FM, we use standard parameter setting provided in the paper [10]: we set all the hyperprior parameters to 1, w_0, w, and v are initialized using a Gaussian distribution with 0 mean and 0.01 variance and all other parameters are set to 0.

TABLE I. TIME COMPARISONS ON DIFFERENT DATASETS.

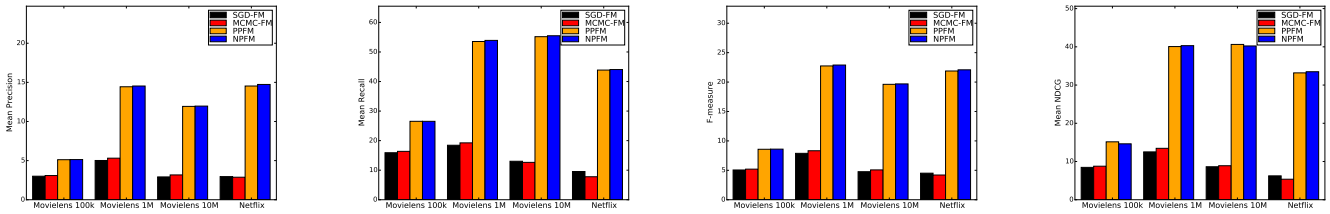
	Time (Hour)			
Datasets	SGD-FM	MCMC-FM	PPFM	NPFM
Movielens 100k	0.00002	0.00003	0.00005	0.00005
Movielens 1m	0.00024	0.00077	0.00077	0.00076
Movielens 10m	0.00442	0.01005	0.00884	0.00871
Netflix	0.05743	0.11330	0.11696	0.11696

B. RESULTS

Fig. 1 shows the results on three Movielens data sets with 100k, 1 million, and 10 million ratings and the Netflix dataset with 100 million ratings. For all the datasets, NPFM and PPFM perform much better than the baseline method SGD-FM and MCMC-FM. We observe that the recall scores (Fig. 1-b) and NDCG values (Fig. 1-d) on all the datasets for NPFM and PPFM are much higher than the other methods. MCMC-FM and SGD-FM perform very poorly for all the datasets primarily due to the inappropriateness of the Gaussian assumption for count data. Table I shows time per iteration for different methods. In all the datasets SGD-FM takes much less time than the other three methods. This is not surprising since SGD-FM is an on-line method and updates model parameters for each data instance, whereas the other three methods are batch algorithms. Though in movielens 100K, MCMC-FM takes less time than the NPFM and PPFM, as dataset size increases, sparsity in the dataset also increases and as a result MCMC-FM, NPFM, and PPFM take almost equal time to run on the two larger datasets. Thus we are able to get the power of Poisson modelling without too much additional computational overhead. What is heartening to note is that NPFM and PPFM perform similarly both in terms of accuracy and time. This indicates that we are able to avoid setting the latent factor dimension apriori but still do not pay much of a cost in terms of performance. This is particularly important when working with datasets where the appropriate dimension is hard to determine ahead of time.

V. CONCLUSION AND FUTURE WORK

This paper describes **N**onparametric **P**oisson **F**actorization **M**achine (NPFM), an alternative for formulating Factorization Machine for count data. The model exploits the natural sparsity of the data, has linear time and space complexity with respect to the number of non-zero observations and predicts the ideal number of latent factors for modeling the pairwise interaction in Factorization Machine. NPFM outperforms some of the



(a) (b) (c) (d)

Fig. 1. Results comparison on different datasets. a) MAP comparison; b) MAR comparison; c) F-measure comparison; d) MNDCG comparison.

existing baselines by significant margin on several real-world datasets. In future, NPFM can be further applied to solving recommendation problems with side information.

VI. ACKNOWLEDGEMENT

This work is supported by the US Office of Naval Reseach grant N00014-14-1-0039, NSF grant IIS-1421729, and Ericsson India research.

REFERENCES

[1] A. Acharya, J. Ghosh, and M. Zhou. Nonparametric Bayesian Factor Analysis for Dynamic Count Matrices. In *Proc. of AISTATS*, pages 1–9. 2015.

[2] A. Acharya, A. Saha, M. Zhou, D. Teffer, and J. Ghosh. Nonparametric Dynamic Network Modeling. In *KDD Workshop on Mining and Learning from Time Series*, 2015.

[3] A. Acharya, D. Teffer, J. Henderson, M. Tyler, M. Zhou, and J. Ghosh. Gamma Process Poisson Factorization for Joint Modeling of Network and Documents. In *Proc. of ECML*, pages 283–299, 2015.

[4] B. Ball, B. Karrer, and M.E.J. Newman. Efficient and principled method for detecting communities in networks. *Phys. Rev. E*, 84, Sep 2011.

[5] D. Blackwell and J. MacQueen. Ferguson distributions via Pólya urn schemes. *The Annals of Statistics*, 1973.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *JMLR*, 3:993–1022, 2003.

[7] A. T. Cemgil. Bayesian inference for nonnegative matrix factorisation models. *Intell. Neuroscience*, 2009.

[8] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011.

[9] T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *Ann. Statist.*, 1973.

[10] C. Freudenthaler, L. Schmidt-Thieme, and S. Rendle. Bayesian factorization machines. In *Proc. of NIPS Workshop on Sparse Representation and Low-rank Approximation*, 2011.

[11] P. Gopalan, J.M. Hofman, and D.M. Blei. Scalable recommendation with poisson factorization. *CoRR*, abs/1311.1704, 2013.

[12] P. Gopalan, F. Ruiz, R. Ranganath, and D. Blei. Bayesian nonparametric poisson factorization for recommendation systems. In *Proc. of AISTATS*, 2014.

[13] P.K. Gopalan, L. Charlin, and D. Blei. Content-based recommendations with poisson factorization. In *Proc. of NIPS*, pages 3176–3184. 2014.

[14] S. Gunasekar, A. Acharya, N. Gaur, and J. Ghosh. Noisy matrix completion using alternating minimization. In *Proc. of ECML PKDD, Part II, LNAI 8189*, pages 194–209, 2013.

[15] T. Joachims. Advances in kernel methods. chapter Making Large-scale Support Vector Machine Learning Practical, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.

[16] N. L. Johnson, A. W. Kemp, and S. Kotz. *Univariate Discrete Distributions*. John Wiley & Sons, 2005.

[17] Y. Kim and S. Choi. Scalable variational bayesian matrix factorization with side information. In *Proc. of AISTATS*, pages 493–502, 2014.

[18] Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proc. of KDD*, pages 426–434, 2008.

[19] Y. Koren. Collaborative filtering with temporal dynamics. In *Proc. of KDD*, pages 447–456, 2009.

[20] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 2009.

[21] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Proc. of NIPS*, 2001.

[22] Y.J. Lim and Y.W. Teh. Variational bayesian approach to movie rating prediction. In *Proc. of KDDCup*, 2007.

[23] R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of computational and graphical statistics*, 2000.

[24] S. Rendle. Factorization machines. In *Proc. of ICDM*, 2010.

[25] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proc. of WWW*, pages 811–820, 2010.

[26] S. Rendle and L. Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proc. of WSDM*, pages 81–90, 2010.

[27] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Proc. of NIPS*, 2007.

[28] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proc. of ICML*, pages 880–887, 2008.

[29] Masa-Aki Sato. Online model selection based on the variational Bayes. *Neural Comput.*, 13(7):1649–1681, July 2001.

[30] J. Silva and L. Carin. Active learning for online bayesian matrix factorization. In *Proc. of KDD*, pages 325–333, 2012.

[31] S. G. Walker. Sampling the Dirichlet mixture model with slices. *Communications in Statistics Simulation and Computation*, 2007.

[32] R. L. Wolpert, M. A. Clyde, and C. Tu. Stochastic expansions using continuous dictionaries: Lévy Adaptive Regression Kernels. *Annals of Statistics*, 2011.

[33] R. L. Wolpert and K. Ickstadt. Poisson/gamma random field models for spatial statistics. *BIOMETRIKA*, 85:251–267, 1998.

[34] L. Xiong, X. Chen, T. Huang, J. Schneider, and J. G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proc. of SDM*, 2010.

[35] M. Zhou. Infinite edge partition models for overlapping community detection and link prediction. In *Proc. of AISTATS*, pages 1135–1143. 2015.

[36] M. Zhou and L. Carin. Augment-and-conquer negative binomial processes. In *Proc. of NIPS*, 2012.

[37] M. Zhou and L. Carin. Negative binomial process count and mixture modeling. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2015.

[38] M. Zhou, L. Hannah, D. Dunson, and L. Carin. Beta-negative binomial process and Poisson factor analysis. In *Proc. of AISTATS*, pages 1462–1471, 2012.

[39] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Proc. 4th Intl Conf. Algorithmic Aspects in Information and Management, LNCS 5034*, 2008.