Bayesian Factorization Machines

Christoph Freudenthaler, Lars Schmidt-Thieme
Information Systems & Machine Learning Lab
University of Hildesheim
31141 Hildesheim
{freudenthaler, schmidt-thieme}@ismll.de

Steffen Rendle
Social Network Analysis
University of Konstanz
78457 Konstanz, Germany
steffen.rendle@uni-konstanz.de

Abstract

This work presents simple and fast structured Bayesian learning for matrix and tensor factorization models. An unblocked Gibbs sampler is proposed for factorization machines (FM) which are a general class of latent variable models subsuming matrix, tensor and many other factorization models. We empirically show on the large Netflix challenge dataset that Bayesian FM are fast, scalable and more accurate than state-of-the-art factorization models.

1 Introduction

Many different problems in machine learning such as computer vision, computational biology or recommender systems deal with complex problems in sparse data. For those problems, e.g., the Netflix prize problem[1], sparse representations such as latent variable models are typically used. The most basic version are matrix factorization models, extensible to tensor factorization models for more than two categorical dimensions. In [5], Rendle generalizes these factorization models to Factorization Machines (FM) and shows how a simple input vector consisting of the appropriate predictors is enough to mimic most factorization models. The general principle is the same as for standard linear regression. For instance factorizing a matrix containing the rating of a user $u \in U$ represented in row $r = 1, \dots, |U|$ for a specific item $i \in I$ represented in column $c = 1, \dots, |I|$ can be generated by defining an input vector of length $p = |U| + |I|$ consisting of p indicator variables. Within the first $|U|$ indicators exactly one indicator is on - the one identifying the current user. Within the last $|I|$ indicators again exactly one is on - now for the current item.

Typically, the number of latent dimensions k for latent variable models is not known. Too small k underfit, too high k overfit the data. Structured Bayesian methods using hierarchical priors overcome brute force grid search for k, e.g., [7, 9] for matrix factorization, [11] for tensor factorization, and [10] multi-relational learning. Each paper derived its own learning algorithm. In this work we come up with *one* structured Bayesian learning algorithm for a much wider class of factorization models – among others subsuming matrix, tensor or multi-relational factorization models. Moreover, the runtime complexity of existing algorithms is typically non-linear in k. A first attempt to tackle this issue is made by Zhu et al. [12]. They adapt the Gibbs sampler of BPMF [7] by unblocking some sampling steps leading to a final runtime complexity of $O(nk + k^2(|U| + |I|))$. In our work, we derive a fast and unblocked Gibbs sampler where each iteration is linear in k. The improvement is based on the work of Rendle et al. [6] where a linear time learning algorithm for FMs using alternating least-squares is presented. As posterior mean and variance of factors are closely related to the (conditional) least-squares solution of [6], we can transfer these ideas to derive fast Bayesian FMs.

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

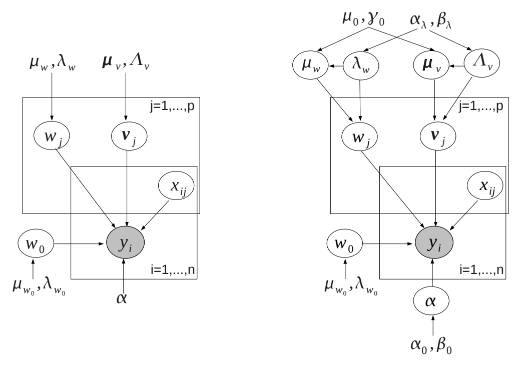

Figure 1: Comparison of the probabilistic interpretation of standard Factorization Machines (left) to Bayesian Factorization Machines (right). BFM extend the standard model by using hyperpriors.

2 Bayesian Factorization Machines

2.1 Factorization Machines (FM)

Factorization Machines [5] are a regression model for a target y using p explanatory variables $\mathbf{x} \in \mathbb{R}^p$ as:

$$y(\mathbf{x}) := w_0 + \sum_{j=1}^{p} w_j \, x_j + \sum_{d=2}^{D} \sum_{j_1=1}^{p} \dots \sum_{j_d > j_{d-1}}^{p} w_{j_1,\dots,j_d} \prod_{l=1}^{d} x_{j_l} \tag{1}$$

where second and higher order parameters $w_{j_1,\dots,j_d}$ are factorized using PARAFAC[2]:

$$w_{j_1,\dots,j_d} := \sum_{f=1}^{k} \prod_{l=1}^{d} v_{j_l,f}^{(d)} \qquad \forall d \geq 2. \tag{2}$$

FM subsume (among others) D-way tensor factorization which is representable by engineering the input features $\mathbf{x}$. Engineering $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_D)$ as a sequence of disjoint subsequences $\mathbf{x}_i = x_{i,1}, \dots, x_{i,p_i}$ where $i = 1, \dots, D$ and $\exists! j \in x_i : x_{i,j} = 1, x_{i,\neg j} = 0 \; \forall i = 1, \dots, D$ gives a D-way tensor factorization model. Note that, FM are more complex than tensor factorization models as $\mathbf{x}$ may consist of real valued predictor variables.

2.2 Bayesian Factorization Machines

So far FM lack Bayesian inference. For obvious reasons such as better accuracy, automatic hyperparameter learning (no grid search) or no learning hyperparameter, e.g., learn rate for gradient descent, we want to enhance FM with structured Bayesian inference. Bayesian theory [1] suggest Bayesian models with hierarchical hyperpriors as they regularize underlying model parameters. The intuition behind this is that additional (hierarchical) *structure* put on model parameters shrinks them towards each other as long as no evidence in the data clearly indicates the opposite. Without loss of generality but for better readability, we present Bayesian FM (BFM) of order $D = 2$.

Our hierarchical Bayesian model adds to the standard regularized regression model Gaussian hyperpriors for each mean μ_θ of all model parameters $\theta \in \Theta = \{w_0, w_1, \dots, w_p, v_{1,1}, \dots, v_{p,k}\}$ but μ_{w_0} as well as Gamma hyperpriors for each prior precision α and λ_θ but λ_{w_0}. The standard L2-regularized regression model with hyperparameters $\Theta_H = \{\lambda_\theta, \mu_\theta : \theta \in \Theta\}$ is for a sample of n observations $(y_i, \mathbf{x}_i) \in \mathbb{R}^{1+p}$:

$$p(\Theta | \mathbf{y}, \mathbf{X}, \Theta_H) \propto \prod_{i=1}^{n} \sqrt{\alpha} \, e^{-\frac{\alpha}{2}(y_i - y(\mathbf{x}_i, \Theta))^2} \prod_{\theta \in \Theta} \sqrt{\lambda_\theta} \, e^{-\frac{\lambda_\theta}{2}(\theta - \mu_\theta)^2} \tag{3}$$

Figure 1 depicts the graphical representations of the standard and our hierarchically extended Bayesian model: we define two regularization populations, i.e. one for all linear effects w_i and another one for the latent variable $v_{1,1}, \dots, v_{p,k}$.

2.3 Inference: Efficient Gibbs Sampling

Since posterior inference for FM is analytically intractable we use Gibbs sampling to draw from the posterior. Standard Gibbs sampling is blocked Gibbs sampling, i.e. all inferred variables Θ and Θ_H b are divided into b disjoint blocks consisting of $r_1, \ldots, r_b$ variables. However, blocked Gibbs sampling for our hierarchical Bayes approach leads to higher time complexities in the final Gibbs sampler as for each block of size r_i matrices need to be inverted which is $O(r_i^3)$. Therefore, we propose single parameter Gibbs sampling. For notational readability, we exploit the multi-linear nature of FM, i.e. for each $\theta \in \Theta$ eq. 1 can be rewritten as [6]:

$$y(\mathbf{x}|\Theta) = g_\theta(\mathbf{x}) + \theta\, h_\theta(\mathbf{x}) \quad \forall \theta \in \Theta \tag{4}$$

where g_θ, h_θ respectively comprise all terms additively, multiplicatively connected to θ (see [6] for details). Using this notation and parameters $\Theta_0 = \{\gamma_0, \mu_0, \alpha_0, \beta_0, \alpha_\lambda, \beta_\lambda\}$ of the conjugate hyperpriors the conditional posterior distributions for each θ, μ_θ, λ_θ, and α resulting from our hierarchical Bayesian model are:

$$p(\theta|\mathbf{y}, \mathbf{X}, \Theta \setminus \{\theta\}, \Theta_H, \Theta_0) = \mathcal{N}(\mu_\theta^*, \sigma_\theta^2), \qquad p(\mu_\theta|\mathbf{y}, \mathbf{X}, \Theta, \Theta_H \setminus \{\mu_\theta\}, \mathbf{\Theta}_0) = \mathcal{N}(\mu_{\mu_\theta}, \sigma_{\mu_\theta}^2)$$
$$p(\lambda_\theta|\mathbf{y}, \mathbf{X}, \Theta, \Theta_H \setminus \{\lambda_\theta\}, \Theta_0) = \Gamma(\alpha_\theta, \beta_\theta), \qquad p(\alpha|\mathbf{y}, \mathbf{X}, \Theta, \Theta_H \setminus \{\alpha\}, \Theta_0) = \Gamma(\alpha_n, \beta_n)$$

with:

$$\mu_\theta^* = \sigma_\theta^2 \left(\sum_{i=1}^{n} \alpha (y_i - g_\theta(\mathbf{x}_i)) h_\theta(\mathbf{x}_i) + \mu_\theta \lambda_\theta \right), \qquad \sigma_\theta^2 = \left(\alpha \sum_{i=1}^{n} h_\theta(\mathbf{x}_i)^2 + \lambda_\theta \right)^{-1} \tag{5}$$

$$\mu_{\mu_\theta} = \sigma_{\mu_\theta}^2\, \lambda_\theta \left(\sum_{j=1}^{p} \theta_j + \gamma_0 \mu_0 \right), \qquad \sigma_{\mu_\theta}^2 = ((p + \gamma_0)\lambda_\theta)^{-1} \tag{6}$$

$$\alpha_\theta = (\alpha_\lambda + p + 1)/2, \qquad \beta_\theta = \left(\sum_{j=1}^{p} (\theta_j - \mu_\theta)^2 + \gamma_0 (\mu_\theta - \mu_0)^2 + \beta_\lambda \right)/2 \tag{7}$$

$$\alpha_n = (\alpha_0 + n)/2, \qquad \beta_n = \left(\sum_{i=1}^{n} (y_i - y(\mathbf{x}_i, \Theta))^2 + \beta_0 \right)/2 \tag{8}$$

The final algorithm iteratively draws samples for hyperparameters Θ_H and model parameters Θ. Sampling of hyperparameters using eq. 6-8 is simple. With a straight-forward implementation it is done in $O(\sum_{i=1}^{n} k\, m(\mathbf{X}_{i.})) = O(k\, N_z)$[2]. Efficient computation of posterior mean and posterior variance of eq. 5 is more complicated. However the posterior mean is identical to the alternating least squares solution for which an efficient algorithm has already proposed [6] with an overall complexity of $O(k\, N_z)$ for an update of all model parameters.

3 Evaluation

We evaluate BFM on the Netflix challenge dataset which is well-known in the recommender systems community. It consists of $n = 100{,}480{,}507$ ratings of $480{,}189$ users for $17{,}700$ movies. The train/test split is the same as in the challenge. This allows direct comparison with existing results reported in [8, 7, 3, 11]. For the parameters of the hyperpriors Θ_0 we have chosen trivial values: $\alpha_0 = \beta_0 = \gamma_0 = \alpha_\lambda = \beta_\lambda = 1, \mu_0 = 0$. Please note, due to the high number of explanatory variables their impact on the Gibbs sampler is negligible (cf. eq. 6-8).

With the derivation of BFM it is possible to learn many different models (e.g. matrix/tensor factorization) Bayesian by appropriate specification of the input data $\mathbf{x}$. Exploiting this flexibility, we tested several models by compiling the feature vector $\mathbf{x}$ from the following components: (1) binary indicators: for each user (u), each item (i), each day of rating (t), each u and t of rating combination (ut), each i and t combination (it)[3], each integer value of the logarithm of the rating user's rating frequency on the day of rating (f), each i and f combination (if), and for items the current user has

[2] $m(\mathbf{X}_{i.})$ is the number of non-zero elements in the i-th row of the design matrix $\mathbf{X}$. $N_z := \sum_{i=1}^{n} m(\mathbf{X}_{i.})$ is the number of non-zero elements in $\mathbf{X}$.

[3] Time is binned in 50 days intervals (following [4]).

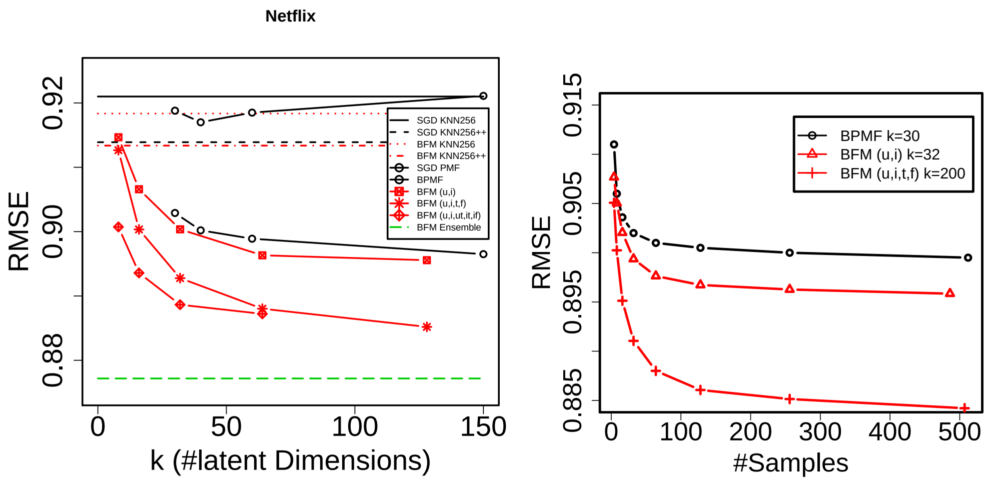

Figure 2: Predictive accuracy of various instances of Bayesian Factorization Machines (BFM) compared to state-of-the-art baselines: SGD PMF [8], BPMF [7], SGD KNN/KNN++ [3]. Right panel: Convergence of unblocked Gibbs sampling (BFM) is comparable to blocked Gibbs sampling (BPMF [7]).

co-rated (++). (2) Real-valued variables: the rating values for the co-rated items (KNN)[4]. We index BFM with the features they use, e.g., BFM (u, i) is a Bayesian matrix factorization model using the user and item indicators u and i. It is similar to BPMF [7]. BFM (u, i, t) uses time information like the Bayesian PARAFAC model BPTF [11]. BFM KNN and BFM KNN++ are the Bayesian equivalent to SGD KNN(++) [3], etc.

Figure 2 plots the quality with a varying amount of factorization dimensions k. As expected, posterior averages in BPMF and BFM (u, i) outperform posterior modes in standard matrix factorization. For the same reason BMF KNN(++) outperform SGD KNN(++). Besides, the Bayesian approaches achieve this without a hyperparameter search. Next, we compare our generic BFM approach to the corresponding specialized models where Gibbs samplers [7, 11] already exist. Comparing BFM (u, i) to BPMF [7], shows that BFM outperforms BPMF slightly. The reason for this are the additional linear effects in BFM. For the time-aware Bayesian factorization model BPTF [11], the best value the authors report is an RMSE value of 0.9044 for $k = 30$. For comparison we also run an BFM (u, i, t) and largely outperform this with 0.8958 ($k = 32$) and 0.8909 ($k = 64$). Moreover, in contrast to BPMF and BPTF using blocked Gibbs, BFM achieve these results with a Gibbs sampler of lower computational complexity. Encouraged by the quality of the single factorization models we also computed a linear ensemble of three BFM: a BFM (u, i, t, f) with $k = 200$, a BFM KNN256++ and a BFM (u, i, ut, it, if) with $k = 64$. The result of this ensembled BFM is 0.8772.

The right panel of figure 2 depicts the convergence rate of blocked[5] and unblocked Gibbs sampling. Surprisingly, the number of iterations for similar models (matrix factorization, $k \approx 30$) using unblocked and blocked Gibbs sampling are comparable.

4 Conclusion

We proposed BFM, a computationally efficient Gibbs sampler for FM. Instead of cubic complexity BFM are linear in k. Empirically tested on the Netflix challenge, BFM converge as fast as models using blocked Gibbs sampling. Thus, BFM combine high accuracy and more scalable learning with the generality of FM which subsume many matrix/tensor factorization models and other (factorized) polynomial regression models like KNN models.

[4]For ++ and KNN we follow the pruning step of [3] and took only the 256 most similar items with respect to the target item.

[5]Parameters are grouped in blocks of size k.

References

[1] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2nd edition, 2003.

[2] Richard A. Harshman. Foundations of the parafac procedure: models and conditions for an 'exploratory' multimodal factor analysis. *UCLA Working Papers in Phonetics*, pages 1–84, 1970.

[3] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, New York, NY, USA, 2008. ACM.

[4] Yehuda Koren. Collaborative filtering with temporal dynamics. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456, New York, NY, USA, 2009. ACM.

[5] Steffen Rendle. Factorization machines. In *Proceedings of the 10th IEEE International Conference on Data Mining*. IEEE Computer Society, 2010.

[6] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2011.

[7] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine Learning*, volume 25, 2008.

[8] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

[9] Mikkel Schmidt. Linearly constrained bayesian matrix factorization for blind source separation. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1624–1632. 2009.

[10] Ilya Sutskever, Ruslan Salakhutdinov, and Joshua Tenenbaum. Modelling relational data using bayesian clustered tensor factorization. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1821–1828. 2009.

[11] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff Schneider, and Jaime G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of the SIAM International Conference on Data Mining (SDM 2010)*, pages 211–222. SIAM, 2010.

[12] Shenghuo Zhu, Kai Yu, and Yihong Gong. Stochastic relational models for large-scale dyadic data using mcmc. In *NIPS*, pages 1993–2000, 2008.