2017 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 25–28, 2017, TOKYO, JAPAN

# MACAU: SCALABLE BAYESIAN FACTORIZATION WITH HIGH-DIMENSIONAL SIDE INFORMATION USING MCMC

*J. Simm*[⋆] *A. Arany*[⋆] *P. Zakeri*[⋆] *T. Haber*[‡] *J. K. Wegner*[†] *V. Chupakhin*[†] *H. Ceulemans*[†] *Y. Moreau*[⋆]

[⋆] ESAT-STADIUS, KU Leuven and imec, B-3001 Heverlee, Belgium
[‡] Hasselt University, 3500 Hasselt, Belgium
[†] Janssen Pharmaceutica, B-2340 Beerse, Belgium

## ABSTRACT

Bayesian matrix factorization is a method of choice for making predictions for large-scale incomplete matrices, due to availability of efficient Gibbs sampling schemes and its robustness to overfitting. In this paper, we consider factorization of large scale matrices with high-dimensional side information. However, sampling the link matrix for the side information with standard approaches costs $O(F^3)$ time, where F is the dimensionality of the features. To overcome this limitation we, firstly, propose a prior for the link matrix whose strength is proportional to the scale of latent variables. Secondly, using this prior we derive an efficient sampler, with linear complexity in the number of non-zeros, $O(N_{nz})$, by leveraging Krylov subspace methods, such as block conjugate gradient, allowing us to handle million-dimensional side information. We demonstrate the effectiveness of our proposed method in drug-protein interaction prediction task.

***Index Terms***— matrix factorization, side information, high scale machine learning, MCMC

## 1. INTRODUCTION

The factorization of *incomplete* matrices is a powerful machine learning tool, having wide range of application in problems where the goal is to have good predictions for the missing values, *e.g.*, recommender systems. It is remarkable that even its Bayesian treatment using efficient Gibbs sampling schemes [1] are able to handle matrices with millions of rows and columns having tens of millions of observed values. Thanks to flexible priors and the use of the posterior for prediction, Bayesian matrix factorization is very robust to overfitting even with large latent dimensions, resulting in improved accuracy while also providing estimates of the uncertainty.

Jaak Simm, Adam Arany, Pooya Zakeri and Yves Moreau are funded by Research Council KU Leuven (CoE PFV/10/016 SymBioSys) and by Flemish Government (IOF, Hercules Stitching, iMinds Medical Information Technologies SBO 2015, IWT: O&O ExaScience Life Pharma; ChemBioBridge, Exaptation, PhD grants).

In this paper we focus on the case where additional features $\mathbf{x}_i \in \mathbb{R}^{F_u}$ are available as side information for rows and $\mathbf{z}_j \in \mathbb{R}^{F_v}$ for columns. For example, consider the task of factorizing a `drug x protein` matrix $\mathbf{Y} \in \mathbb{R}^{N_u \times N_v}$ with less than 1% observed values. This is very challenging task because almost all drugs have only few observations, and thus, their latent vectors are poorly estimated resulting in inaccurate predictions. To overcome this problem we can use the side information regarding the compounds to guide the factorization.

However, the most informative features for compounds are high-dimensional. To tackle such problems, we propose MCMC-based factorization method *Macau* that is able to scale to millions of features, $F_u >$ 1,000,000 while the factorized matrix $\mathbf{Y}$ can have a million of rows (and/or columns). In contrast, the recent sampling-based Bayesian factorization methods Bayesian Matrix Factorization with Side Information (BMFSI) [2] and Bayesian Canonical PARAFAC with Features and Networks (BCPFN) [3] have complexity $O(F_u^3)$ because they require explicit computation and Cholesky decomposition of the covariance matrix of size $F_u \times F_u$ for linking the side information to the factorization.

In Macau, the side information $\mathbf{x}_i$ is linked to the latent vectors $\mathbf{u}_i \in \mathbb{R}^D$ in a regression-like fashion by link matrix $\boldsymbol{\beta}_u \in \mathbb{R}^{F_u \times D}$, where $D$ is the number of latent dimensions. The good scalability of Macau comes from the use of a novel *noise injection sampler* for $\boldsymbol{\beta}_u$, which has linear complexity in terms of the number of non-zeros in the side information matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_{N_u}]^\top$.

This paper makes following contributions. Firstly, we propose a prior for the link matrix whose strength is proportional to the scale of latent variables allowing us to derive the noise injection sampler (see Section 2.2). Secondly, to derive this sampler we introduce *noise injection trick* that converts the problem of sampling the link matrix to solving a specially designed linear system with noise added into the right-hand sides (RHSs) (see Section 3.1). To solve this system for high-dimensional settings we propose to use Conjugate Gradient and Block Conjugate Gradient methods, which are well established Krylov subspace methods (see Section 3.2 and 3.3).

978-1-5090-6341-3/17/$31.00 ©2017 IEEE

Finally, in Section 5, we demonstrate Macau on a challenging task of drug-protein activity prediction.

## 2. MATRIX FACTORIZATION WITH SIDE INFORMATION

In this section we introduce the model used by Macau, including the scale-adjusted prior for the link matrix $\boldsymbol{\beta}$.

### 2.1. General Setup

The main idea in matrix factorization is to represent each row and each column by latent vectors $\mathbf{u}_i, \mathbf{v}_j \in \mathbb{R}^D$ where the prediction of the model is given by $\hat{Y}_{ij} = \mathbf{u}_i^\top \mathbf{v}_j$. The observation noise is assumed to be Gaussian, as was proposed in Bayesian Probabilistic Matrix Factorization [1],

$$p(\mathbf{Y}|\mathbf{U}, \mathbf{V}, \alpha) = \prod_{(i,j)\in I} \mathcal{N}(Y_{ij}|\mathbf{u}_i^\top \mathbf{v}_j, \alpha^{-1}), \qquad (1)$$

where $\mathcal{N}$ is the normal distribution, $I$ is the set of matrix cells whose value has been observed and $\alpha \in \mathbb{R}_+$ is the noise precision.

We place a common multivariate Gaussian prior on the latent vectors of the rows

$$p(\mathbf{u}_i|\boldsymbol{\Lambda}_u, \boldsymbol{\mu}_u, \mathbf{x}_i, \boldsymbol{\beta}_u) = \mathcal{N}(\mathbf{u}_i|\boldsymbol{\mu}_u + \boldsymbol{\beta}_u^\top \mathbf{x}_i, \boldsymbol{\Lambda}_u^{-1}) \qquad (2)$$

$$p(\boldsymbol{\mu}_u, \boldsymbol{\Lambda}_u|\Theta_0) = \mathcal{NW}(\boldsymbol{\mu}_u, \boldsymbol{\Lambda}_u|\Theta_0), \qquad (3)$$

where $\boldsymbol{\mu}_u \in \mathbb{R}^D$ and $\boldsymbol{\Lambda}_u \in \mathbb{R}^{D\times D}$ are the shared intercept vector and precision matrix for all rows, $\boldsymbol{\beta}_u \in \mathbb{R}^{F_u \times D}$ is the link matrix for rows and $\Theta_0$ are the fixed hyperparameters for the Normal-Wishart distribution. The term $\boldsymbol{\beta}_u^\top \mathbf{x}_i$ allows the prior mean of $\mathbf{u}_i$ to depend on the side information $\mathbf{x}_i$ and, therefore, have informed predictions also for rows with very few or no measurements. We use analogous prior for the columns.

Without the $\boldsymbol{\beta}_u^\top \mathbf{x}_i$ in the prior (2) our model is equivalent to Bayesian Probabilistic Matrix Factorization [1]. Originally, the adjustment of the prior mean was first proposed for non-Bayesian models [4]. Recently, it was used in MCMC-based sparse tensor factorization in BCPFN [3] and has been shown to be the best performing option for incorporating the side information in the context of Variational Bayes factorization methods [5].

Additionally, there is a recent variational method called VBMFSI-CA [6]. It is able to incorporate *high-dimensional* side information by approximating the posterior distribution of $\boldsymbol{\beta}$ by a product of univariate normally distributed factors, *i.e.*, $q(\boldsymbol{\beta}) = \prod_{f=1}^{F} \prod_{d=1}^{D} \mathcal{N}(\beta_{fd}|\mu_{fd}, \sigma_{fd}^2)$. This allows VBMFSI-CA to scale linearly with the number of non-zeros, similarly to the method proposed in this paper. However, in contrast to our MCMC strategy, this univariate approximation have a strong negative impact on the predictive performance as can be seen in the experiments (see Section 5).

### 2.2. Scale-adjusted Prior on the Link Matrix

The standard approach in the previous works, for example used by Rai et al. [3], has been to add a zero-mean Gaussian prior on $\boldsymbol{\beta}_u$, *i.e.*, $\mathrm{vec}(\boldsymbol{\beta}_u) \sim \mathcal{N}(\mathbf{0}, \lambda_{\boldsymbol{\beta}_u}^{-1}\mathbf{I})$, where $\mathbf{I}$ is the identity matrix and $\lambda_{\boldsymbol{\beta}_u} \in \mathbb{R}_+$ is the precision value.

Instead, we propose a prior whose strength is proportional to the scale of $\boldsymbol{\Lambda}_u$, making the effect of the prior *invariant* to the scale of the latent vectors:

$$p(\boldsymbol{\beta}_u|\boldsymbol{\Lambda}_u, \lambda_{\boldsymbol{\beta}_u}) = \mathcal{N}(\mathrm{vec}(\boldsymbol{\beta}_u)|\mathbf{0}, (\boldsymbol{\Lambda}_u \otimes \lambda_{\boldsymbol{\beta}_u}\mathbf{I}_{F_u})^{-1}), \quad (4)$$

where $\otimes$ is the Kronecker product. The reason for introducing this prior is that $\boldsymbol{\beta}_u$ acts as a regression weight matrix whose inputs $\mathbf{x}_i$ are fixed but whose outputs $\mathbf{u}_i$ are a moving target (changing over the Gibbs iterations). Moreover, the prior has the advantage of being well suited for sampling, as we show in the next section.

Finally, as the value of $\lambda_{\boldsymbol{\beta}_u}$ is data dependent we infer it by introducing a fixed Gamma hyperprior $\lambda_{\boldsymbol{\beta}_u} \sim \mathcal{G}(\mu_0, \nu_0)$. The graphical model of Macau is depicted in Figure 1, where both rows and columns have side information.

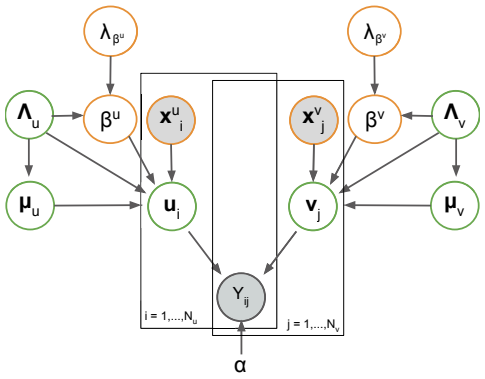


**Fig. 1**. Graphical model of Macau (without the fixed hyperprior parameters). The observed variables are shaded.

## 3. NOISE INJECTION SAMPLER

We use Gibbs sampler for inference in the model, which we fully explain in Section 4. In this section we focus on the main idea of sampling the link matrix $\boldsymbol{\beta}_u$ in the settings with high-dimensional side information (large $F_u$) and large number of rows in $\mathbf{Y}$ (large $N_u$).

The main idea of our noise injection sampler is that we will construct a specific linear system whose solution corresponds to a sample from the conditional distribution of $\boldsymbol{\beta}_u$. For large $F_u$ the linear system of size $F_u \times F_u$ will be solved without explicitly computing the system matrix using *Krylov subspace methods*, which only require (sparse) matrix-vector product operations.

### 3.1. Noise Injection Trick

The conditional posterior of $\boldsymbol{\beta}_u$, used for Gibbs sampler, combines (2) and (4). Due to the scale invariant effect of the prior (4) we get

$$p(\boldsymbol{\beta}_u|\ldots) \propto \exp\left(-\frac{1}{2}\operatorname{tr}\left[(L_u + \lambda_{\boldsymbol{\beta}_u}\boldsymbol{\beta}_u^\top\boldsymbol{\beta}_u)\boldsymbol{\Lambda}_u\right]\right) \quad (5)$$

$$\text{where } L_u = (\tilde{\mathbf{U}} - \mathbf{X}\boldsymbol{\beta}_u^\top)^\top(\tilde{\mathbf{U}} - \mathbf{X}\boldsymbol{\beta}_u^\top) \quad (6)$$

where for compactness we have dropped the conditional terms after $\boldsymbol{\beta}_u$ and use shorthands $\tilde{\mathbf{U}} = [\mathbf{u}_1 - \boldsymbol{\mu}_u, \ldots, \mathbf{u}_{N_u} - \boldsymbol{\mu}_u]^\top$ and $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_{N_u}]^\top$. The scale invariance of the prior (4) allowed us to factorize out $\boldsymbol{\Lambda}_u$ in (5) (the derivation is given in an extended version). This gives us the Gaussian mean and precision of $\boldsymbol{\beta}_u$ as

$$p(\boldsymbol{\beta}_u|\ldots) = \mathcal{N}(\mathbf{K}^{-1}\mathbf{X}^\top\tilde{\mathbf{U}}, \boldsymbol{\Lambda}_u^{-1} \otimes \mathbf{K}^{-1}) \quad (7)$$

$$\text{where } \mathbf{K} = \mathbf{X}^\top\mathbf{X} + \lambda_{\boldsymbol{\beta}_u}\mathbf{I}. \quad (8)$$

Naively sampling from Gaussian (7) would cost $O({F_u}^3 D^3)$ time. However, due to the special structure of the Gaussian distribution (7) we can generate its sample by solving the following $F_u \times F_u$ linear systems with $D$ right-hand sides:

$$\mathbf{K}\tilde{\boldsymbol{\beta}}_u = \mathbf{X}^\top(\tilde{\mathbf{U}} + \mathbf{E}_1) + \sqrt{\lambda_{\boldsymbol{\beta}_u}}\mathbf{E}_2 \quad (9)$$

with respect to $\tilde{\boldsymbol{\beta}}_u$, where $\mathbf{E}_1 \in \mathbb{R}^{N_u \times D}$ and $\mathbf{E}_2 \in \mathbb{R}^{F_u \times D}$ are *noise* matrices with each row drawn from $\mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}_u^{-1})$. We call (9) the *noise injection trick* as it adds noise to the right-hand sides of the linear systems. The reduction of the size of the problem from $F_u D \times F_u D$ to $F_u \times F_u$ is thanks to the decoupling $\mathbf{K}$ and $\boldsymbol{\Lambda}_u$. It is straightforward to see the correctness of the noise injection trick by deriving the mean and covariance of $\tilde{\boldsymbol{\beta}}_u$ from (9) and noting its identity to (7).

### 3.2. Iterative Noise Injection Sampler

The immediate advantage of formulating the sampling of $\boldsymbol{\beta}_u$ as solving the linear systems is that we can tap into the rich field of numerical methods. If the linear system is medium dimensional, $F_u < 20{,}000$, it is fastest to precompute $\mathbf{X}^\top\mathbf{X}$, then during Gibbs iterations compute $\mathbf{K}$ by adding the diagonal terms and solve (9) using direct solvers, like Cholesky or QR decomposition. This sampler has computational complexity of $O(F_u^3 + DF_u^2 + DF_u N_u)$. If $F_u$ and $N_u$ are in the same order of magnitude the term $O(F_u^3)$ dominates the complexity.

In the case of high-dimensional side information using direct solvers becomes intractably expensive. Since the matrix $\mathbf{K}$ is positive definite a good option for solving the systems is to use *linear conjugate gradient* (CG), a Krylov subspace method well established for solving large scale linear systems with good convergence properties. An alternative is to use Block-CG, which we discuss in the Section 3.3. Krylov subspace methods do not require direct access to the matrix $\mathbf{K}$ but only require access to its linear operator $\mathbf{K}(\mathbf{r})$ [7]. As CG solves a single system with one RHS we will use $D$ separate CGs each solving a different RHS column of (9). This also allows us to easily parallelize the sampler by running each CG in parallel.

**Algorithm 1** Iterative noise injection sampler

**Require:** $\mathbf{X}$, $\lambda_{\boldsymbol{\beta}_u}$, $\tilde{\mathbf{U}}$, $\boldsymbol{\Lambda}_u$ and tolerance $\epsilon$
**Ensure:** $\boldsymbol{\beta}_u$ is a sample from (7)
1: Sample $\mathbf{E}_1$ by drawing $N_u$ vectors from $\mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}_u^{-1})$
2: Sample $\mathbf{E}_2$ by drawing $F_u$ vectors from $\mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}_u^{-1})$
3: Define operator $\mathbf{K}(\mathbf{r}) \leftarrow \mathbf{X}^\top(\mathbf{X}\mathbf{r}) + \lambda_{\boldsymbol{\beta}_u}\mathbf{r}$
4: $\mathbf{B} \leftarrow \mathbf{X}^\top(\tilde{\mathbf{U}} + \mathbf{E}_1) + \sqrt{\lambda_{\boldsymbol{\beta}_u}}\mathbf{E}_2$
5: Solve $\mathbf{K}(\boldsymbol{\beta}_u) = \mathbf{B}$ for $\boldsymbol{\beta}_u$ using Block-CG (or CG) with tolerance $\epsilon$
6: Return $\boldsymbol{\beta}_u$

The use of CG allows the proposed noise injection sampler to handle side information $\mathbf{X}$ with millions of sparse features $F_u$ as the linear operator $\mathbf{K}(\mathbf{r})$ can be implemented by two products of sparse-matrix and dense-vector, and an addition:

$$\mathbf{K}(\mathbf{r}) = \mathbf{K}\mathbf{r} = \mathbf{X}^\top(\mathbf{X}\mathbf{r}) + \lambda_{\boldsymbol{\beta}_u}\mathbf{r}. \quad (10)$$

Firstly, the RHSs of (9) need to be computed and then the iterative solver can be executed. The iterative noise injection sampler is described in Alg. 1. The computational complexity of the product $\mathbf{K}\mathbf{r}$ is $O(N_{nz} + F_u + N_u)$, where $N_{nz}$ is the number of non-zeros in the matrix $\mathbf{X}$. Although the number of CG iterations required for convergence depends on the condition number of $\mathbf{K}$, it does not directly depend on its size, therefore, the total complexity of sampling $D$ RHSs is $O(DN_{nz} + DF_u + DN_u)$.

Another aspect of the iterative sampler Alg. 1 is that it has an error tolerance as an input parameter that controls the stopping criteria for the iterative method. As the RHSs have noise injected into them it is not critical to demand overly exact solutions. In our experiments even the error tolerances of $10^{-4}$ or $10^{-5}$ resulted in similar final (test) prediction accuracies as tolerance $10^{-6}$. In the reported experiments we used $10^{-6}$ but in practice $10^{-4}$ should be a safe choice for the tolerance, reducing the computation time by 30-40%.

In addition to sparse side information $\mathbf{X}$, the iterative noise injection sampler is also useful for dealing with high-dimensional *dense* side information when the number of rows in matrix $\mathbf{Y}$ is small, i.e., $N_u \ll F_u$.

### 3.3. Block Conjugate Gradient

An even better option besides CG is to use its multiple RHSs version called *block conjugate gradient* (Block-CG) [8]. Block-CG, instead of working in the standard Krylov space

span($\mathbf{r}_0, \mathbf{K}\mathbf{r}_0, \mathbf{K}^2\mathbf{r}_0, \ldots$) where $\mathbf{r}_0$ is the initial vector, operates in the block Krylov space span($\mathbf{R}_0, \mathbf{K}\mathbf{R}_0, \mathbf{K}^2\mathbf{R}_0, \ldots$), where $\mathbf{R}_0$ is the initial residual *matrix*. The main benefit of Block-CG, versus solving all RHSs in separate CGs, is that while the computation effort per iteration is slightly increased, the number of iterations required is reduced [9]. It is theoretically known that the reduction factor is between 1 and $1/D$. The reason for this is that $D$ residual vectors jointly span the Krylov subspace. For more detailed discussion of block Krylov subspace methods see [10].

In our drug dataset with 105k-dimensional $\mathbf{X}$ (described in more detail in Section 5) we observed that using Block-CG decreased the number of iterations (red line on Figure 2) by 2x for 8 RHSs and by 3.5x for 32 RHSs, resulting in speedups of 2.8x for 8 RHS and 3.8x for 32 RHSs for the computation time per RHS in a single threaded Block-CG setup. The computation times are shown in blue on Figure 2, where they are scaled to match the number of iterations at $N_{rhs} = 1$. These results show that the additional computation per iteration is only a minor overhead. Moreover, the total computation time per RHS actually drops faster than the number of iterations (see Figure 2), which is due to the increased arithmetic density in the sparse-matrix-dense-matrix-product $\mathbf{KR}$, resulting in better CPU cache behavior.

We also ran the setup with different tolerance $\epsilon$ and precision $\lambda_{\beta_u}$ and observed similar gains from using more RHSs.

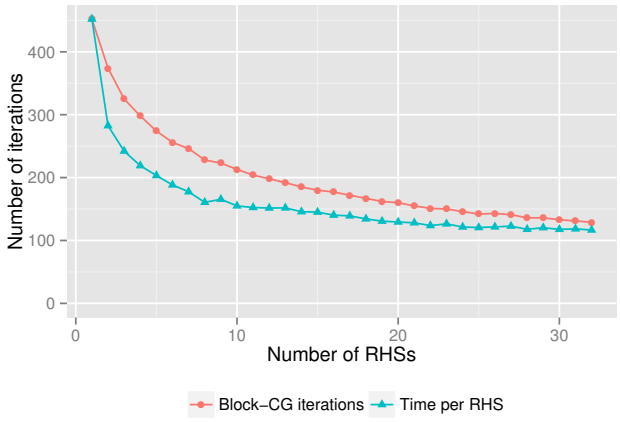


**Fig. 2**. The number of required Block-CG iterations (red) and the scaled computation time per RHS vs the number of RHSs for 105k dimensional sparse $\mathbf{X}$ for tolerance $10^{-6}$ and precision $\lambda_{\beta_u} = 2.5$.

The exact steps of Block-CG are outlined in Alg. 2. It is easy to see that if the RHS matrix $\mathbf{B}$ has just a single column, Block-CG is equivalent to CG. Similarly to CG, almost all of the effort in an iteration is the execution of the operator $\mathbf{KP}_k$. The extra computational effort incurred by Block-CG is the computation and solving of two small dense linear systems in each iteration (on line 7 in Alg. 2 and on line 14 in Alg. 2). As can be seen from Figure 2, this extra effort is negligible, and therefore, the reduction in the number of iterations directly translates into speed-up.

**Algorithm 2** Block conjugate gradient

**Require:** Operator $\mathbf{K}$, RHSs $\mathbf{B}$, initial guess $\boldsymbol{\beta}_0$, toler. $\epsilon$
**Ensure:** $\mathbf{K}\boldsymbol{\beta}^* = \mathbf{B}$
1: **Initialize:**
2: $\quad \bar{\mathbf{R}}_0 = \mathbf{B} - \mathbf{K}\boldsymbol{\beta}_0$
3: $\quad$ Orthogonalize and normalize $\bar{\mathbf{R}}_0 = \mathcal{Q}\mathcal{R}$ (using QR)
4: $\quad$ Set $\mathbf{P}_0 \leftarrow \mathcal{Q}$, $\mathcal{X}_0 \leftarrow 0$ and $\mathbf{R}_0 \leftarrow \mathcal{Q}$
5: **for** $k = 0, 1, 2, \ldots$ **do**
6: $\quad$ // Solution update:
7: $\quad \mathbf{A}_k \leftarrow (\mathbf{P}_k^\top \mathbf{K}\mathbf{P}_k)^{-1}\mathbf{R}_k^\top \mathbf{R}_k$
8: $\quad \mathcal{X}_{k+1} \leftarrow \mathcal{X}_k + \mathbf{P}_k\mathbf{A}_k$
9: $\quad \mathbf{R}_{k+1} \leftarrow \mathbf{R}_k - \mathbf{K}\mathbf{P}_k\mathbf{A}_k$
10: $\quad$ // Convergence check:
11: $\quad$ **if** $\|\mathbf{R}_{k+i}^{(j)}\| < \epsilon$, the column $j$ exits the iteration
12: $\quad$ **if** all columns exited, **break**
13: $\quad$ // Search subspace construction:
14: $\quad \Psi_k \leftarrow (\mathbf{R}_k^\top \mathbf{R}_k)^{-1}\mathbf{R}_{k+1}^\top \mathbf{R}_{k+1}$
15: $\quad \mathbf{P}_{k+1} \leftarrow \mathbf{R}_{k+1} + \mathbf{P}_k\Psi_k$
16: **end for**
17: Return $\boldsymbol{\beta}^* = \boldsymbol{\beta}_0 + \mathcal{X}_{k+1}\mathcal{R}$

The computational complexity of Block-CG is $O((DN_{nz} + D^2F_u + DN_u)\gamma(D))$ where $1 \geq \gamma(D) \geq \frac{1}{D}$ is the reduction factor for Block-CG mentioned before. Comparing to the complexity $O(DN_{nz} + DF_u + DN_u)$ of the CG-based sampler, the gain is due to the factor $\gamma(D)$.

### 3.4. Parallelization

In the single node multi-core setup it is straightforward to parallelize separate CGs by just running them independently in parallel. In the case of Block-CG we parallelize the sparse-matrix-dense-matrix products for $\mathbf{X}$ and $\mathbf{X}^\top$. In our implementation we split both $\mathbf{X}$ and $\mathbf{X}^\top$ into row blocks that then can be multiplied in parallel with a dense matrix using OpenMP parallelism[1].

Table 1 shows the computational performance of the iterative noise injection sampler on high-dimensional sparse side information for drugs with 32 RHSs using a single 36-core node[2]. The matrix $\mathbf{X}$ contains features for 168k drugs from ChEMBL repository [11] and we also report computation times for subsets. Note that in the case of the subsets we removed unused features from $\mathbf{X}$, resulting in smaller $F_u$. As the table shows the total computation time scales almost linearly with respect to $N_{nz}$. It is interesting to note that the number of required Block-CG iterations increases with $N_u$

[1] C/C++ code is available on Github: https://github.com/jaak-s/macau
[2] The system has 2 Intel Haswell Xeon E5-2699 v3 CPUs @ 2.3GHz.

**Table 1**. Computational performance of iterative noise injection sampler on $\boldsymbol{\beta}_u$ with $D = 32$, using a single node with 36 threads. The tolerance $\epsilon$ is $10^{-6}$ and precision $\lambda_{\boldsymbol{\beta}_u} = 5.0$.

| $N_u$ | $F_u$ | $N_{nz}$ | ITERATIONS | TIME |
|---|---|---|---|---|
| 50K | 166K | 3.5M | 165 | 22S |
| 100K | 246K | 7.3M | 227 | 48S |
| 150K | 282K | 11.0M | 280 | 75S |
| 168K | 292K | 12.2M | 299 | 82S |

but is balanced out by the fact that the parallelization is becoming more efficient for larger $\mathbf{X}$, resulting in almost linear scaling. The increase in the number of Block-CG iterations is due to the increase in the condition number of the matrix $\mathbf{K}$ as the sparsity of the matrix $\mathbf{X}$ increases (due to the fact that the number of non-zeros per row is constant while $F_u$ is growing).

## 4. MACAU

As already discussed Macau uses block Gibbs sampling, where the scalable sampling of the link matrices $\boldsymbol{\beta}_u$ and $\boldsymbol{\beta}_v$ was described in Section 3. The sampling of the other variables, namely, the latent vectors $\mathbf{u}_i$,$\mathbf{v}_j$, the Gaussian priors $\boldsymbol{\mu}_u$, $\boldsymbol{\Lambda}_u$,$\boldsymbol{\mu}_v$, $\boldsymbol{\Lambda}_v$, and the precisions $\lambda_{\boldsymbol{\beta}_u}$, $\lambda_{\boldsymbol{\beta}_v}$ is easy to derive thanks to the conjugacy of the priors. Namely, $\mathbf{u}_i$ is sampled from $\mathcal{N}(\boldsymbol{\mu}_i^*, [\boldsymbol{\Lambda}_i^*]^{-1})$ with

$$\boldsymbol{\Lambda}_i^* = \boldsymbol{\Lambda}_u + \alpha \sum_{j \in I_u(i)} \mathbf{v}_j \mathbf{v}_j^\top,$$

$$\boldsymbol{\mu}_i^* = [\boldsymbol{\Lambda}_i^*]^{-1} \left( \boldsymbol{\Lambda}_u(\boldsymbol{\mu}_u + \boldsymbol{\beta}_u^\top \mathbf{x}_i) + \alpha \sum_{j \in I_u(i)} Y_{ij} \mathbf{v}_j \right),$$

where $I_u(i)$ is the set of columns $j$ for which $Y_{ij}$ has been observed. The sampling of $\mathbf{v}_j$ is analogous. The sampling of prior variables $\boldsymbol{\mu}_u$, $\boldsymbol{\Lambda}_u$, $\boldsymbol{\mu}_v$, $\boldsymbol{\Lambda}_v$ follows BPMF [1]. We also sample the precision variables $\lambda_{\boldsymbol{\beta}_u}$ and $\lambda_{\boldsymbol{\beta}_v}$ for the link matrices from the gamma distribution. We found out it is advisable to set the initial $\lambda_{\boldsymbol{\beta}_u}$ and $\lambda_{\boldsymbol{\beta}_v}$ to conservative (not too small) values to avoid high condition number of $\mathbf{K}$ in the beginning of the burn-in phase.

As the latent vectors $\mathbf{u}_i$ are conditionally independent given $\mathbf{V}$, $\boldsymbol{\mu}_u$, and $\boldsymbol{\Lambda}_u$ we can sample them in parallel, allowing Macau to scale to $\mathbf{Y}$ with millions of rows and columns on modern multi-core CPUs.

## 5. EXPERIMENTS

In this section, we compare Macau to related methods on the problem of drug-protein activity prediction, which has high-dimensional side information on drugs. Specifically, we compare Macau to BPMF [1], separate ridge regression for each protein, scalable variational approach VBMFSI-CA [6], and a general factorization library libFM [12].

**Table 2**. Results for Drug-Protein activity prediction experiments.

| METHOD | RMSE | NEGLL |
|---|---|---|
| BPMF (D=30) | $0.895 \pm 0.007$ | $1.225 \pm 0.008$ |
| VBMFSI-CA (D=30) | $0.802 \pm 0.011$ | $1.168 \pm 0.014$ |
| RIDGE REGRESSION | $0.741 \pm 0.008$ | NA |
| LIBFM (D=30) | $0.651 \pm 0.007$ | NA |
| MACAU (D=30) | $\mathbf{0.612 \pm 0.005}$ | $\mathbf{0.876 \pm 0.005}$ |

The prediction of drug and protein interactions is crucial for the development of new drugs. In our case studies we focus on the interaction measure IC50, which measures the concentration of the drug necessary to inhibit the activity of the protein by $50\%$. We prepared a data set from the public bioactivity database ChEMBL [11] Version 19. First, we selected proteins that had at least 200 IC50 measurements, and then we kept drugs with 3 or more IC50 measurements. The final numbers for small molecules and proteins are 15,073 and 346, respectively, with total of 59,280 IC50 measurements. In all of the experiments we model $\log_{10}$ of IC50 and set the precision $\alpha = 5.0$, because this corresponds to a reasonable standard deviation of $0.45$[3]. For drugs, we use sparse features (substructure fingerprint) with $F_{drug} = 105{,}672$, and for proteins, we use dense features (based on protein sequence) with $F_{prot} = 20$.

We use 400 burn-in iterations and 1600 posterior samples, the experiments are repeated 10 times. All factorization methods use 30 latent dimensions. To tune the regularization parameter of the ridge regression of each protein, we used 5-fold inner cross-validation. In every run we randomly sample 20% of the observations into a test set.

As it can be seen from Table 2, Macau significantly outperforms all related methods, both in terms of root mean square error (RMSE) and negative log-likelihood (negLL). Although VBMFSI-CA uses a very similar model its performance is considerably worse compared to Macau, which implies that the univariate variational approximation used by VBMFSI-CA has a severe negative impact.

## 6. DISCUSSION

There is a general iterative sampling method called conjugate direction sampler (CD-sampler) [13] that generates a sample from $\mathcal{N}(0, \mathbf{A}^{-1})$ by following *noisy* conjugate gradient like steps. However, CD-sampler has known convergence problems in practice and is recommended only to be used as an initial approximation, requiring further refinement by variable-wise Gibbs sampler [13]. On the other hand, our proposed

[3] It is possible to enhance the performance by tuning/sampling $\alpha$.

*noise injection sampler* is specialized for the linear regression setting and results in solving a linear system. Its convergence properties come directly from the underlying linear solver used (either Cholesky, CG or Block-CG). Moreover, applying the CD-sampler to the conditional distribution (7) of $\boldsymbol{\beta}_u$ has a significant overhead due to the need to iterate over $\boldsymbol{\Lambda}_u \otimes \mathbf{K}$. In comparison, our proposed sampler uses the noise injection trick to decouple $\mathbf{K}$ and $\boldsymbol{\Lambda}_u$, therefore, making the system smaller (from one $F_u D \times F_u D$ to $D$ systems of size $F_u \times F_u$) and also more easily parallelizable.

The sampling of the link matrix $\boldsymbol{\beta}_u$ is related to the sampling of weights $\mathbf{w} \in \mathbb{R}^F$ in Bayesian linear regression

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{i=1}^{N} \mathcal{N}(y_i|\mathbf{w}^\top \mathbf{x}_i, \tau^{-1}) \tag{11}$$

$$p(\mathbf{w}, \lambda|\nu, \mu) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda^{-1}\mathbf{I})\, \mathcal{G}(\lambda|\nu, \mu), \tag{12}$$

where we assume $\mathbf{w}$ contains the bias term. In Bayesian linear regression we want to infer the posterior distribution of $\mathbf{w}$ (and distribution of $\lambda$), but in the high-dimensional case explicitly computing the precision matrix and then sampling $\mathbf{w}$ is intractable. It is possible to leverage the proposed noise injection sampler to sample $\mathbf{w}$ by solving the linear system

$$\mathbf{C}\tilde{\mathbf{w}} = \tau \mathbf{X}^\top (\mathbf{y} + \mathbf{e}_1) + \mathbf{e}_2, \tag{13}$$

$$\text{where } \mathbf{C} = \tau \mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_F, \tag{14}$$

where $\mathbf{e}_1 \sim \mathcal{N}(\mathbf{0}, \tau^{-1}\mathbf{I}_N)$ and $\mathbf{e}_2 \sim \mathcal{N}(\mathbf{0}, \lambda \mathbf{I}_F)$ are randomly generated vectors. The proof is presented in the extended version. Analogously to Alg. 1 we can use CG to solve the linear system in the case of high-dimensional $\mathbf{X}$.

## 7. CONCLUSION

We have proposed Macau, a Bayesian factorization method for large scale matrices with high-dimensional side information. To make the inference efficient we presented a noise injection sampler based on a scale-adjusted prior on the link matrix. The sampling can exploit modern Krylov subspace based solvers such as CG and Block-CG that have good convergence properties. Finally, we demonstrated the effectiveness of our method on a drug-protein activity prediction dataset, where the drugs have high-dimensional side information.

## 8. REFERENCES


[1] Ruslan Salakhutdinov and Andriy Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 880–887.

[2] Ian Porteous, Arthur U Asuncion, and Max Welling, "Bayesian matrix factorization with side information and dirichlet process mixtures.," in *AAAI*, 2010.

[3] Piyush Rai, Yingjian Wang, and Lawrence Carin, "Leveraging features and networks for probabilistic tensor decomposition," in *AAAI Conference on Artificial Intelligence*, 2015.

[4] Deepak Agarwal and Bee-Chung Chen, "Regression-based latent factor models," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 19–28.

[5] Sunho Park, Yong-Deok Kim, and Seungjin Choi, "Hierarchical bayesian matrix factorization with side information," in *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. AAAI Press, 2013, pp. 1593–1599.

[6] Yong-Deok Kim and Seungjin Choi, "Scalable variational bayesian matrix factorization with side information," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS), Reykjavik, Iceland*, 2014.

[7] Jorge Nocedal and Stephen Wright, *Numerical optimization*, Springer Science & Business Media, 2006.

[8] Dianne P O'Leary, "The block conjugate gradient algorithm and related methods," *Linear algebra and its applications*, vol. 29, pp. 293–322, 1980.

[9] YT Feng, DRJ Owen, and D Perić, "A block conjugate gradient method applied to linear systems with multiple right-hand sides," *Computer methods in applied mechanics and engineering*, vol. 127, no. 1, pp. 203–215, 1995.

[10] Martin H Gutknecht, "Block krylov space methods for linear systems with multiple right-hand sides: an introduction," 2006.

[11] A Patrícia Bento, Anna Gaulton, Anne Hersey, Louisa J Bellis, Jon Chambers, Mark Davies, Felix A Krüger, Yvonne Light, Lora Mak, Shaun McGlinchey, et al., "The chembl bioactivity database: an update," *Nucleic acids research*, vol. 42, no. D1, pp. D1083–D1090, 2014.

[12] Steffen Rendle, "Factorization machines with libfm," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 3, no. 3, pp. 57, 2012.

[13] Colin Fox, "Polynomial accelerated mcmc and other sampling algorithms inspired by computational optimization," in *Monte Carlo and Quasi-Monte Carlo Methods 2012*, pp. 349–366. Springer, 2013.