Fast Bayesian Non-Negative Matrix Factorisation and Tri-Factorisation

Supplementary Materials

Thomas Brouwer, Jes Frellsen, Pietro Lio’
University of Cambridge

Submitted to NIPS 2016 Workshop – Advances in Approximate Bayesian Inference.

Contents

1 Model details  2
   1.1 Gibbs sampling for matrix factorisation . . . . . . . . . . . . . . . . . . . . . . . 2
   1.2 Variational Bayes for matrix factorisation . . . . . . . . . . . . . . . . . . . . . 3
   1.3 BNMTF Gibbs sampling parameter values . . . . . . . . . . . . . . . . . . . . . 3
   1.4 BNMTF Variational Bayes parameter updates . . . . . . . . . . . . . . . . . . . 4

2 Model discussion  6
   2.1 Complexity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
   2.2 Model selection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
   2.3 Initialisation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
   2.4 Implementation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
   2.5 Code . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

3 Additional experiments  9
   3.1 Model selection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
   3.2 Missing values . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
   3.3 Noise test . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
   3.4 Drug sensitivity predictions . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
1 Model details

1.1 Gibbs sampling for matrix factorisation

In this section we offer an introduction to Gibbs sampling, and show how it can be applied to the Bayesian non-negative matrix factorisation model.

Gibbs sampling works by sampling new values for each parameter \( \theta \) from its marginal distribution given the current values of the other parameters \( \theta_{-i} \), and the observed data \( D \). If we sample new values in turn for each parameter \( \theta_i \) from \( p(\theta_i|\theta_{-i}, D) \), we will eventually converge to draws from the posterior, which can be used to approximate the posterior \( p(\theta|D) \).

We have to discard the first \( n \) draws because it takes a while to converge (burn-in), and since consecutive draws are correlated we only use every \( i \)th value (thinning).

For the Bayesian non-negative matrix factorisation model this means that we need to be able to draw from the following distributions:

\[
p(U_{ik}|\tau, U_{-ik}, V, D) \quad p(V_{jk}|\tau, U, V_{-jk}, D) \quad p(\tau|U, V, D)
\]

where \( U_{-ik} \) denotes all elements in \( U \) except \( U_{ik} \), and similarly for \( V_{-jk} \). Using Bayes theorem we can obtain the posterior distributions. For example, for \( p(U_{ik}|\tau, U_{-ik}, V, D) \):

\[
p(U_{ik}|\tau, U_{-ik}, V, D) \propto p(D|\tau, U, V) \times p(U_{ik}|\lambda^U_{ik})
\]

\[
\propto \prod_{j \in \Omega_i} N(R_{ij}|U_i \cdot V_j, \tau^{-1}) \times E(U_{ik}|\lambda^U_{ik})
\]

\[
\propto \exp\left\{ \frac{\tau}{2} \sum_{j \in \Omega_i} (R_{ij} - U_i \cdot V_j)^2 \right\} \times \exp\left\{ -\lambda^U_{ik} U_{ik} \right\} \times u(x)
\]

\[
\propto \exp\left\{ \frac{U_{ik}^2}{2} \left[ \tau \sum_{j \in \Omega_i} V_{jk}^2 \right] + U_{ik} \left[ -\lambda^U_{ik} + \tau \sum_{j \in \Omega_i} (R_{ij} - \sum_{k' \neq k} U_{ik'} V_{jk'}) V_{jk} \right] \right\} \times u(x)
\]

\[
\propto \mathcal{T}N(U_{ik}|\mu^U_{ik}, \tau^U_{ik})
\]

where

\[
\mathcal{T}N(x|\mu, \tau) = \begin{cases} \sqrt{\frac{\tau}{2\pi}} \exp\left\{ -\frac{1}{2} \left( x - \mu \right)^2 \right\} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}
\]

is a truncated normal: a normal distribution with zero density below \( x = 0 \) and renormalised to integrate to one. \( \Phi(\cdot) \) is the cumulative distribution function of \( \mathcal{N}(0, 1) \).

Applying the same technique to the other posteriors gives us:

\[
p(\tau|U, V, D) = \mathcal{G}(\tau|\alpha^*, \beta^*)
\]

\[
p(V_{jk}|\tau, U, V_{-jk}, D) = \mathcal{T}N(V_{jk}|\mu^V_{jk}, \tau^V_{jk})
\]

The parameters of these distributions are given in Table 1 where \( \Omega_i = \{ j \mid (i, j) \in \Omega \} \) and \( \Omega_j = \{ i \mid (i, j) \in \Omega \} \).
1.2 Variational Bayes for matrix factorisation

For the Variational Bayes algorithm for inference, updates for the approximate posterior distributions are given in Table 1, and were obtained using the techniques described in the paper. We use \( E_q[f(X)] \) as a shorthand for \( E_q[f(X)] \), where \( X \) is a random variable and \( f \) is a function over \( X \). We make use of the identity \( \tilde{X}^2 = \tilde{X}^2 + \text{Var}_q[X] \). The expectation and variance of the parameters with respect to \( q \) are given below, for random variables \( X \sim G(a, b) \) and \( Y \sim \mathcal{TN}(\mu, \tau) \).

\[
\tilde{X} = \frac{a}{b}, \quad \tilde{Y} = \mu + \frac{1}{\sqrt{\tau}} \lambda (-\mu \sqrt{\tau}), \quad \text{Var}[Y] = \frac{1}{\tau} [1 - \delta (-\mu \sqrt{\tau})]
\]

where \( \psi(x) = \frac{d}{dx} \log \Gamma(x) \) is the digamma function, \( \lambda(x) = \phi(x)/[1 - \Phi(x)] \), and \( \delta(x) = \lambda(x)[\lambda(x) - x] \). \( \phi(x) = \frac{1}{\sqrt{2\pi}} \exp\{-\frac{1}{2} x^2\} \) is the density function of \( \mathcal{N}(0, 1) \).

1.3 BNMTF Gibbs sampling parameter values

For the BNMTF Gibbs sampling algorithm, we sample from the following posteriors:

\[
p(\tau|F, S, G, D) = \mathcal{G}(\tau|\alpha^*, \beta^*)
\]
\[
p(F_{ik}|\tau, F_{-ik}, S, G, D) = \mathcal{TN}(F_{ik}|\mu^F_{ik}, \tau^F_{ik})
\]
\[
p(S_{kl}|\tau, F, S_{-kl}, G, D) = \mathcal{TN}(S_{kl}|\mu^S_{kl}, \tau^S_{kl})
\]
\[
p(G_{jl}|\tau, F, S, G_{-jl}, D) = \mathcal{TN}(G_{jl}|\mu^G_{jl}, \tau^G_{jl})
\]
The updates for the parameters are given in Table 2 below.

Table 2: NMTF Gibbs Update Rules

<table>
<thead>
<tr>
<th>GIBBS SAMPLING</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha^*$</td>
<td>$\alpha + \frac{1}{2}</td>
</tr>
<tr>
<td>$\beta^*$</td>
<td>$\beta + \frac{1}{2} \sum_{(i,j) \in \Omega} (R_{ij} - F_i \cdot S \cdot G_j)^2$</td>
</tr>
<tr>
<td>$\tau_{ik}^F$</td>
<td>$\tau \sum_{j \in \Omega} (S_k \cdot G_j)^2$</td>
</tr>
<tr>
<td>$\mu_{ik}^F$</td>
<td>$\frac{1}{\tau_{ik}} \left( -\lambda_{ik}^F + \tau \sum_{j \in \Omega} \left( R_{ij} - \sum_{k' \neq k} \sum_{l=1}^L F_{ik'} S_{k'j} G_{jl} \right) (S_k \cdot G_j) \right)$</td>
</tr>
<tr>
<td>$\tau_{kl}^S$</td>
<td>$\tau \sum_{(i,j) \in \Omega} F_{ik} G_{jl}^2$</td>
</tr>
<tr>
<td>$\mu_{kl}^S$</td>
<td>$\frac{1}{\tau_{kl}} \left( -\lambda_{kl}^S + \tau \sum_{(i,j) \in \Omega} \left( R_{ij} - \sum_{k' \neq k, l' \neq l} F_{ik'} S_{k'j} G_{jl'} \right) F_{ik} G_{jl} \right)$</td>
</tr>
<tr>
<td>$\tau_{jl}^G$</td>
<td>$\tau \sum_{i \in \Omega_j} (F_i \cdot S_{.,l})^2$</td>
</tr>
<tr>
<td>$\mu_{jl}^G$</td>
<td>$\frac{1}{\tau_{jl}} \left( -\lambda_{jl}^G + \tau \sum_{i \in \Omega_j} \left( R_{ij} - \sum_{k=1}^K \sum_{l' \neq l} F_{ik} S_{kl} G_{jl'} \right) (F_i \cdot S_{.,l}) \right)$</td>
</tr>
</tbody>
</table>

1.4 BNMTF Variational Bayes parameter updates

As discussed in the paper, the term $\mathbb{E}_q \left[ (R_{ij} - F_i \cdot S \cdot G_j)^2 \right]$ adds extra complexity to the matrix tri-factorisation case.

$$\mathbb{E}_q \left[ (R_{ij} - F_i \cdot S \cdot G_j)^2 \right] =$$

$$\left( R_{ij} - \sum_{k=1}^K \sum_{l=1}^L \tilde{F}_{ik} \tilde{S}_{kl} \tilde{G}_{jl} \right)^2$$

$$+ \sum_{k=1}^K \sum_{l=1}^L \text{Var}_q [F_{ik} S_{kl} G_{jl}]$$

(1)

$$+ \sum_{k=1}^K \sum_{l=1}^L \sum_{k' \neq k} \text{Cov}_q [F_{ik} S_{kl} G_{jl}, F_{ik'} S_{kl'} G_{jl'}]$$

(2)

$$+ \sum_{k=1}^K \sum_{l=1}^L \sum_{l' \neq l} \text{Cov}_q [F_{ik} S_{kl} G_{jl}, F_{ik'} S_{kl'} G_{jl'}]$$

(3)
The above variance and covariance terms are equal to the following, respectively:

$$\widetilde{F}_{ik}^2 \widetilde{S}_{kl}^2 \widetilde{G}_{jl}^2 - \widetilde{F}_{ik}^2 \widetilde{S}_{kl}^2 \widetilde{G}_{jl}^2$$  \hspace{1cm} (1)

$$\text{Var}_q [F_{ik}] S_{kl} G_{jl} S_{kl} G_{jl}$$  \hspace{1cm} (2)

$$\widetilde{F}_{ik} S_{kl} \text{Var}_q [G_{jl}] \widetilde{F}_{ik} S_{kl}$$  \hspace{1cm} (3)

The updates for the variational parameters of the Variational Bayes algorithm for the Bayesian non-negative matrix tri-factorisation are given in Table 3 below.

<table>
<thead>
<tr>
<th>TABLE 3: NMTF VB Update Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha^*$</td>
</tr>
<tr>
<td>$\beta^*$</td>
</tr>
<tr>
<td>$\tau_{ik}^F$</td>
</tr>
<tr>
<td>$\mu_{ik}^F$</td>
</tr>
<tr>
<td>$\tau_{kl}^S$</td>
</tr>
<tr>
<td>$\mu_{kl}^S$</td>
</tr>
<tr>
<td>$\tau_{jl}^G$</td>
</tr>
<tr>
<td>$\mu_{jl}^G$</td>
</tr>
</tbody>
</table>


## 2 Model discussion

### 2.1 Complexity

The updates for the Gibbs samplers and VB algorithms can be implemented efficiently using matrix operations. The time complexity per iteration for Bayesian non-negative matrix factorisation is $O(IJK^2)$ for both Gibbs and VB, and $O(IJ(K^2L + KL^2))$ per iteration for tri-factorisation. However, the updates in each column of $U, V, F, G$ are independent of each other and can therefore be updated in parallel.

For the Gibbs sampler, this means we can draw these values in parallel, but for the VB algorithm we can jointly update the columns using a single matrix operation. Modern computer architectures can exploit this using vector processors, leading to a great speedup.

Furthermore, after the VB algorithm converges we have our approximation to the posterior distributions immediately, whereas with Gibbs we need to obtain further draws after convergence and use a thinning rate to obtain an accurate estimate of the posterior. This deterministic behaviour of VB makes it easier to use. Although additional variables need to be stored to represent the posteriors, this does not result in a worse space complexity, as the Gibbs sampler needs to store draws over time.

### 2.2 Model selection

In practice we do not know the optimal model dimensionality of our data, and we need to estimate its value. In our case, we want to find the best value of $K$ for matrix factorisation, and $K, L$ for tri-factorisation.

The log probability of the data, $\log p(D|\theta)$, is a good measure for the quality of the fit to the data. As $K$ increases we expect the log likelihood to improve as we give the model more freedom for fitting to the data, but this can lead to overfitting. Therefore, we need to penalise the model’s performance by its complexity. We use the Akaike information criterion (AIC) [Akaike 1974] defined as

$$AIC = 2k - 2\log p(D|\theta)$$

where $k$ is the number of free parameters in our model. For matrix factorisation this is $IK + JK$ and for tri-factorisation $IK + KL + JL$.

Another popular measure is the Bayesian information criterion (BIC) [Schwarz 1978]. BIC tends to penalise complicated models more heavily than AIC. We found that AIC peaked closer to the true model dimensionality on synthetic data than BIC, especially for matrix tri-factorisation, and we therefore use the former.

For matrix factorisation we then try different values for $K$ in a given range and pick the $K$ that gives the lowest AIC. Similarly for matrix tri-factorisation, we can perform a grid search for a range of values for $K$ and $L$, trying each possible $(K, L)$ pair, but this results in training $K \times L$ different models. Instead, we can perform a greedy search on the grid, as illustrated in Figure [1].
• We are given a grid of values \((K_i, L_i)\).

• We start with the lowest values, \((K_1, L_1)\), train a model, and measure the model quality.

• For each of the three points above it – \((K_i, L_{i+1})\), \((K_{i+1}, L_i)\), \((K_{i+1}, L_{i+1})\) – we train a model and measure the model quality.

• The model that gives the best improvement is selected as our next value on the grid. If no improvement is made, the current point \((K_i, L_i)\) gives the best values for \(K\) and \(L\).

![Diagram of Greedy Search Procedure for Model Selection]

Since we are looking for the best fitting model, we can train multiple models with random initialisations for each \(K, L\) and use the one with the highest log likelihood (we denote restarts).

### 2.3 Initialisation

Initialising the parameters of the models can vastly influence the quality of convergence. This can be done for example by using the hyperparameters \(\lambda_{ik}^U, \lambda_{jk}^Y, \lambda_{ik}^F, \lambda_{ik}^S, \lambda_{jk}^G, \alpha, \beta\) to set the initial values to the mean of the priors of the model. Alternatively, we can use random draws of the priors as the initial values. We found that random draws tend to give faster and better convergence than the expectation.

For matrix tri-factorisation we can also initialise \(F\) by running the K-means clustering algorithm on the rows as datapoints, and similarly \(G\) for the columns, as suggested by Ding et al. [2006]. For the VB algorithm we then set the \(\mu\) parameters to the cluster indicators, and for Gibbs we set the matrices to the cluster indicators, plus 0.2 for smoothing. We found that this improved the convergence as well, with \(S\) initialised using random draws.

### 2.4 Implementation

All algorithms mentioned were implemented using the Python language. The numpy package was used for fast matrix operations, and for random draws of the truncated normal distribution we used the Python package rtnorm by C. Lassner [http://miv.u-strasbg.fr/mazet/rtnorm/], giving more efficient draws than the standard libraries and dealing with rounding errors.
The mean and variance of the truncated normal involve operations prone to numerical errors when $\mu \ll 0$. To deal with this we observe that when $\mu \sqrt{\tau} \ll 0$ the truncated normal distribution approximates an exponential one with rate $\mu \sqrt{\tau}$, and therefore has mean $1/(\mu \sqrt{\tau})$ and variance $1/(\mu \sqrt{\tau})^2$.

All experiments were run on a Medion Erazer laptop with an Intel i7-3610QM CPU (4 cores of 2.30GHz each), GeForce GTX 670M graphics card, and 16GB memory.

2.5 Code

Implementations of all discussed methods are available online, via [https://github.com/ThomasBrouwer/BNMTF/](https://github.com/ThomasBrouwer/BNMTF/).
3 Additional experiments

3.1 Model selection

To demonstrate our proposed model selection framework (see section 2.2) we use the toy dataset described earlier, using our VB algorithms. We let each model train for 1000 iterations with 5 restarts.

As can be seen in Figure 2a, the mean square error on the training data for matrix factorisation converges after $K = 10$, whereas Figure 2b shows that using the Akaike information criterion gives a clear peak at the true $K$. The ELBO also provides a good heuristic for model selection, as seen in figure 2c.

Figure 2e shows the full grid search for matrix tri-factorisation, and gives a peak at $K = 4, L = 4$. This is slightly lower than the true $K, L$, but shows that a good enough fit can be achieved using fewer factors. The proposed greedy search in Figure 2f finds the same solution but only trying 13 of the 100 possible combinations, suggesting that this model selection procedure can offer a significant speedup with similar performance.

We also ran the model selection frameworks on the drug sensitivity dataset, where the true number of latent factors are unknown. Figure 3 shows that for matrix factorisation the best value for $K$ is around 25, and for matrix tri-factorisation $K = L = 5$.

![Figure 2: Model selection for VB-NMF (top row) and VB-NMTF (bottom row). We measure the model quality for different values of $K$ (and $L$) on the toy datasets. The true $K$ for NMF is 10, and the true $K, L$ for NMTF is 5, 5. Figures 2a–2c show that the MSE cannot find the right model dimensionality for NMF, but the AIC and ELBO can. The same applies to NMTF, as shown in Figures 2d–2f, where we additionally see that the proposed greedy search model selection method finds the same solution as the full grid one, but trying only 13 of the 100 possible values.](image-url)
3.2 Missing values

We furthermore tested the ability of our model to recover missing values as the fraction of unknown entries increases (more sparse datasets). We run each algorithm on the same dataset for 1000 iterations (burn-in 800, thinning rate 5) to give the algorithms enough time to converge, splitting the data randomly ten times each into test and training data, and computing the average mean square error of the predictions on the test data.

High errors are indicative of overfitting or not converging to a good solution. We can see in Figure 4a that the fully Bayesian methods for matrix factorisation obtain good predictive power even at 70% missing values, whereas ICM starts failing there. The non-probabilistic method starts overfitting from 20% missing values, leading to very high prediction errors.

For matrix tri-factorisation we notice that our VB method sometimes does not converge to the best solution for 50% or more missing values. This is shown in Figure 4b. As a result, the average error is higher than the other methods in those cases.

3.3 Noise test

We conducted a noise test to measure the robustness of the different methods. Our experiment works in a similar manner to the missing values test, but now adding different levels of Gaussian noise to the data, and with 10% test data. The noise-to-signal ratio is given by the ratio of the variance of the Gaussian noise we add, to the variance of the generated data. We see in Figures 4c and 4d that the non-probabilistic approach starts overfitting heavily at low levels of noise, whereas the fully Bayesian approaches achieve the best possible predictive powers even at high levels of noise.

3.4 Drug sensitivity predictions

Finally, we performed cross-validation experiments on three different drug sensitivity experiments. Firstly, the Genomics of Drug Sensitivity in Cancer (GDSC v5.0, Yang et al. [2013]) dataset contains 138 drugs and 622 cell lines, with 81% of entries observed (as introduced in the paper). Secondly, the Cancer Cell Line Encyclopedia (CCLE, Barretina et al. [2012]) has 504 drugs and 22 cell lines. There are two versions: one detailing IC_{50} drug sensitivity
values (96% observed) and another giving $EC_{50}$ values (63% observed).

We compare our methods against classic algorithms for matrix factorisation and tri-factorisation. Aside from the Gibbs sampler (G-NMF, G-NMTF) and VB algorithms (VB-NMF, VB-NMTF), we consider the non-probabilistic matrix factorisation (NP-NMF) and tri-factorisation (NP-NMTF) methods introduced by Lee and Seung [2000] and Yoo and Choi [2009], respectively. Schmidt et al. [2009] also proposed an Iterated Conditional Modes (ICM-NMF) algorithm for computing an MAP solution, where instead of using draws from the posteriors as updates we set their values to the mode. We also extended this method for matrix tri-factorisation (ICM-NMTF).

For the GDSC dataset we also compare with a recent paper by Ammad-ud din et al. [2014] which uses a method called Kernelised Bayesian Matrix Factorisation (KBMF), leveraging similarity kernels of the drugs and cell lines. We reconstructed the drug kernels using targets, PaDeL fingerprints, and 1D and 2D descriptors. Similarly for the cell lines we used gene expression, copy-number variation, and cancer mutation data. For the other datasets we only compared the matrix factorisation models.

The results of running 10-fold cross-validation can be found in Table 4. For KBMF and non-probabilistic NMF and NMTF we use nested cross-validation to find the best value for $K$ (and $L$). For the other methods we use cross-validation with the model selection detailed in the supplementary materials (Section 2.2).

We can see the Gibbs sampling NMF model performs the best in two of the three datasets, outperforming even the KBMF model which uses side information. The ICM models tend to overfit to the data, and often led to very high predictive errors. The non-probabilistic models do well on the large GDSC dataset, but less so on the small CCLE datasets with only 24 rows. The Bayesian models do significantly better on these two.

The matrix tri-factorisation models generally perform as well as its matrix factorisation counterpart. For matrix factorisation, the fast VB version does worse than the Gibbs sampling variant. However, for matrix tri-factorisation VB outperforms Gibbs on two of the three datasets.
Table 4: 10-fold cross-validation drug sensitivity prediction results (mean squared error). Very high predictive errors are replaced by $\infty$, and the best performances are highlighted in bold.

<table>
<thead>
<tr>
<th></th>
<th>NMF</th>
<th></th>
<th></th>
<th>NMTF</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>KBMF</td>
<td>NP</td>
<td>ICM</td>
<td>G</td>
<td>VB</td>
<td>NP</td>
</tr>
<tr>
<td>GDSC IC$_{50}$</td>
<td>2.144</td>
<td>2.251</td>
<td>$\infty$</td>
<td>2.055</td>
<td>2.282</td>
<td>2.258</td>
</tr>
<tr>
<td>CCLE IC$_{50}$</td>
<td>-</td>
<td>4.683</td>
<td>$\infty$</td>
<td>3.719</td>
<td>3.984</td>
<td>4.664</td>
</tr>
<tr>
<td>CCLE EC$_{50}$</td>
<td>-</td>
<td>8.047</td>
<td>$\infty$</td>
<td>7.807</td>
<td>7.805</td>
<td>8.076</td>
</tr>
</tbody>
</table>

Bibliography


