Bayesian Boolean Matrix Factorisation

Tammo Rukat [1]   **Chris C. Holmes** [1 2]   **Michalis K. Titsias** [3]   **Christopher Yau** [4]

Abstract

Boolean matrix factorisation aims to decompose a binary data matrix into an approximate Boolean product of two low rank, binary matrices: one containing meaningful patterns, the other quantifying how the observations can be expressed as a combination of these patterns. We introduce the OrMachine, a probabilistic generative model for Boolean matrix factorisation and derive a Metropolised Gibbs sampler that facilitates efficient parallel posterior inference. On real world and simulated data, our method outperforms all currently existing approaches for Boolean matrix factorisation and completion. This is the first method to provide full posterior inference for Boolean Matrix factorisation which is relevant in applications, e.g. for controlling false positive rates in collaborative filtering and, crucially, improves the interpretability of the inferred patterns. The proposed algorithm scales to large datasets as we demonstrate by analysing single cell gene expression data in 1.3 million mouse brain cells across 11 thousand genes on commodity hardware.

1. Introduction

Boolean matrix factorisation (BooMF) can infer interpretable decompositions of a binary data matrix $\boldsymbol{X} \in \{0, 1\}^{N \times D}$ into a pair of low-rank, binary matrices $\boldsymbol{Z} \in \{0, 1\}^{N \times L}$ and $\boldsymbol{U} \in \{0, 1\}^{D \times L}$. The data generating process is based on the Boolean product, a special case of matrix product between binary matrices where all values

[1]Department of Statistics, University of Oxford, UK [2]Nuffield Department of Medicine, University of Oxford, UK [3]Department of Informatics, Athens University of Economics and Business, Greece [4]Centre for Computational Biology, Institute of Cancer and Genomic Sciences, University of Birmingham, UK. Correspondence to: Tammo Rukat <tammo.rukat@stats.ox.ac.uk>.
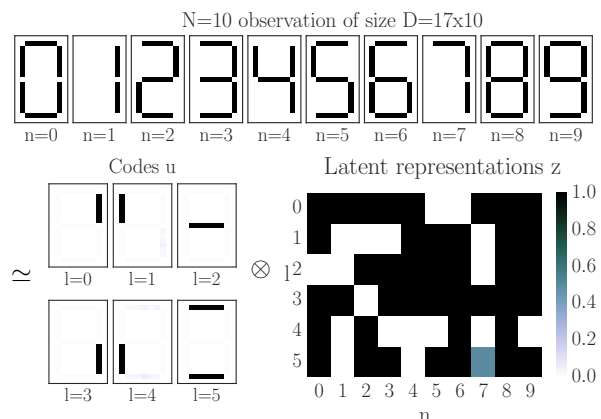
Figure 1. The observed images are 10 digits from 0 to 9 as they are traditionally represented in calculators. The data is factorised into matrices of rank 6, which is not sufficient for full error-free reconstruction. Every digit, except 7, can be constructed by Boolean combination of the inferred codes. The OrMachine infers a posterior mean probability of 50% for using code $l = 5$ in constructing a 7. Note that there exist other equally valid solutions to this problem with 6 latent dimensions. The pixels represent posterior means. Codes and observations are arranged to 10×17 images for interpretation.

larger than zero are set to one, i.e.

$$x_{nd} = \bigvee_{l=1}^{L} z_{nl} \wedge u_{ld} . \qquad (1)$$

Here, $\vee$ and $\wedge$ encode the Boolean disjunction and conjunction, respectively. BooMF provides a framework for learning from binary data where the inferred codes $\boldsymbol{U}$ provide a basis and the indicator variables $\boldsymbol{Z}$ encode the presence or absence of these codes. This representation is illustrated in the calculator digits example in Fig. 1. We can think of BooMF as binary factor analysis or as clustering with joint assignments, where each observation is assigned to a subset of L cluster centroids or codes. The L-dimensional indicators provide a compact representation of which codes are allocated to each observation. As stated in Eq. (1), a feature x_{nd} takes a value of one if it equals one in any of the assigned codes.

BooMF has many real-world applications ranging from topic modelling (Blei, 2012) to collaborating filtering (Su

& Khoshgoftaar, 2009) and computer vision (Lázaro-Gredilla et al., 2016). In this paper, we introduce the Or-Machine, a Bayesian approach to BooMF, and fit the model using a fast and scalable Metropolised Gibbs sampling algorithm. On simulated and real-world data, our method is shown to significantly outperform the current state-of-the-art message passing approaches for learning BooMF models. Moreover, we consider a challenging application in the analysis of high-throughput single cell genomics data. BooMF is used to identify latent gene signatures (codes) that correspond to key cellular pathways or biological processes from large gene expression datasets consisting of 1.3 million cells across 11 thousand genes. Genes are expressed if one or more relevant biological processes are active, a property which is naturally modelled by the Boolean OR operation. We also introduce a multi-layered extensions of Bayesian BooMF that can capture hierarchical dependencies in the latent representations.

2. Related Work

There has been a sustained interest in BooMF and related methods of which we will give a brief review. The Discrete Basis Problem (Miettinen et al., 2006) provides a greedy heuristic algorithm to solve BooMF without recourse to an underlying probabilistic model. It is based on association rule mining (Agrawal et al., 1994) and has more recently been extended to automatically select the optimal dimensionality of the latent space based on the minimum description length principle (Miettinen & Vreeken, 2014). In contrast, multi assignment clustering for Boolean data (Streich et al., 2009) leverages on a probabilistic model for BooMF, adding a further global noise source to the generative process. Point estimates are inferred by deterministic annealing. Similarly, Wood et al. (2012) develop a probabilistic model to infer hidden causes. In contrast to the Boolean OR, the likelihood of an observation increases with the number of active hidden codes. They use an Indian Buffet process prior over the latent space and a Gibbs sampler to infer the distribution over the unbounded number of hidden causes. A more expressive model for matrix factorisation with binary latent variables is introcued by (Meeds et al., 2007), who combine binary interactions with continous weights. A similar approach to ours is the work by Ravanbakhsh et al. (2016). The authors tackle BooMF using a probabilistic graphical model and derive a message passing algorithm to perform MAP inference. Their method is shown to have state-of-the-art performance for BooMF and completion. It therefore serves us as baseline benchmark in these tasks. The message passing approach has recently been employed by Lázaro-Gredilla et al. (2016) in a hierarchical network combined with pooling layers to infer the building blocks of binary images.

3. The OrMachine

3.1. Model Formulation

The OrMachine is a probabilistic generative model for Boolean matrix factorisation. A matrix of N binary observations $\boldsymbol{x}_n \in \{0,1\}^D$ is generated from a discrete mixture of L binary codes $\mathbf{u}_l \in \{0,1\}^D$. Binary latent variables z_{nl} denote whether or not code l is used in generating a particular observation $\boldsymbol{x}_n$. The probability for a data point x_{nd} to be one is greater than $1/2$ if the corresponding codes and latent variables in at least one latent dimension both equal one; conversely, if there exists no dimension where codes and latent variables both equal one, the probability for the data point to be one is less than $1/2$. The exact magnitude of this probability is inferred from the data and, for later notational convenience, is parametrised as the logistic sigmoid of a global dispersion parameter $\sigma(\lambda) = (1+e^{-\lambda})^{-1}$, with $\lambda \in \mathbb{R}^+$. Next, we give a full description of the likelihood and prior distributions used in the OrM.

The likelihood function is factorised across the N observations and D features with each factor given by

$$p(x_{nd}|\boldsymbol{u}, \boldsymbol{z}, \lambda) = \begin{cases} \sigma(\lambda); & \text{if } x = \min(1, \boldsymbol{u}_d^T \boldsymbol{z}_n) \\ 1 - \sigma(\lambda); & \text{if } x \neq \min(1, \boldsymbol{u}_d^T \boldsymbol{z}_n) \end{cases} \quad (2)$$

$$= \sigma \left[\lambda \tilde{x}_{nd} \left(1 - 2 \prod_l (1 - z_{nl} u_{ld}) \right) \right]. \quad (3)$$

Tilde denotes the $\{0,1\} \rightarrow \{-1,1\}$ mapping so that for any binary variable $x \in \{0,1\}$, $\tilde{x} = 2x - 1$. The expression inside the parentheses of Eq. (3) encodes the OR operation and evaluates to 1 if $z_{nl} = u_{ld} = 1$ for at least one l, and to -1 otherwise. The dispersion parameter controls the noise in the generative process, i.e. as $\lambda \rightarrow \infty$, all probabilities tend to 0 or 1 and the model describes a deterministic Boolean matrix product. Note that the likelihood can be computed efficiently from Eq. (3) as we describe in detail in the next section. We further assume independent Bernoulli priors for all variables u_{ld} and z_{nl}. Such priors allow us to promote denseness or sparsity in codes and latent variables. Notice that the designation of $\boldsymbol{U}$ as codes and $\boldsymbol{Z}$ as latent variables is not necessary since these matrices appear in a symmetric manner. If we transpose the matrix of observations $\boldsymbol{X}$, then codes and latent variables merely swap roles.

Finally, we do not place a prior on the dispersion parameter λ, but maximise it using an EM-type algorithm described below.

3.2. Fast Posterior Inference

The full joint distribution of all data and random variables is given by $p(\boldsymbol{X}, \boldsymbol{U}, \boldsymbol{Z}|\lambda) = p(\boldsymbol{X}|\boldsymbol{U}, \boldsymbol{Z}, \lambda)p(\boldsymbol{U})p(\boldsymbol{Z})$.

The full conditional for z_{nl} (and analogous for u_{ld}) is

$$p(z_{nl}|\cdot) =$$
$$\sigma\left[\lambda\tilde{z}_{nl}\sum_d\tilde{x}_{nd}u_{ld}\prod_{l'\neq l}(1-z_{nl'}u_{l'd})+\text{logit}(p(z_{nl}))\right]. \quad (4)$$

We give a detailed derivation in the supplement. Notice that the independent Bernoulli prior enters the expression as additive term inside the sigmoid function that vanishes for the uninformative Bernoulli prior $p(z) = 1/2$.

The form of Eq. (4) allows for computationally efficient evaluation of the conditionals. The underlying principle is that once certain conditions are met, the result of the full conditional is known without considering the remainder of a variable's Markov blanket. For instance, when computing updates for z_{nl}, terms in the sum over d necessarily evaluate to zero if one of the following conditions is met: (i) $u_{ld} = 0$ or (ii) $z_{nl'}u_{l'd} = 1$ for some $l' \neq l$. This leads to Algorithm 1 for fast evaluation of the conditionals.

Algorithm 1 Computation of the full conditional of z_{nl}

accumulator $= 0$
for d in $1, \ldots, D$ **do**
  if $u_{ld} = 0$ **then**
    continue (next iteration over d)
  end if
  for l' in $1, \ldots, L$ **do**
    if $l' \neq l$ and $z_{nl'} = 1$ and $u_{l'd} = 1$ **then**
      continue (next iteration over d)
    end if
  end for
  accumulator $=$ accumulator $+ \tilde{x}_{nd}$
end for
$p(z_{nl}|\cdot) = \sigma\left(\lambda \cdot \tilde{z}_{nl} \cdot \text{accumulator}\right)$

To infer the posterior distribution over all variables u_{ld} and z_{nl} we could iteratively sample from the above conditionals using standard Gibbs sampling. In practice we use a modification of this procedure which is referred to as Metropolised Gibbs sampler and was proposed by Liu (1996). We always propose to flip the current state, leading to a Hastings acceptance probability of $p(z|\cdot)/(1-p(z|\cdot))$. This is guaranteed to yield lower variance Monte Carlo estimates (Peskun, 1973).

After every sweep through all variables, the dispersion parameter λ is updated to maximise the likelihood akin to the M-step of a Monte Carlo EM algorithm. Specifically, given the current values of the codes $\boldsymbol{U}$ and latent variables $\boldsymbol{Z}$ we can compute how many observations x_{nd} are correctly predicted by the model, as
$P = \sum_{n,d}\mathbb{I}\left[x_{nd} = 1 - \prod_l(1 - z_{nl}u_{ld})\right]$. This allows us to

---

Algorithm 2 Sampling from the OrMachine

for i in $1, \ldots, \text{max-iters}$ **do**
  for n in $1, \ldots, \text{N}$ *(in parallel)* **do**
    for l in $1, \ldots, \text{L}$ **do**
      Compute $p(z_{nl}|\cdot)$ following Algorithm 1
      Flip z_{nl} with probability $[p(z_{nl}|\cdot)^{-1}-1]^{-1}$
    end for
  end for
  for d in $1, \ldots, \text{d}$ *(in parallel)* **do**
    for l in $1, \ldots, \text{L}$ **do**
      Compute $p(u_{ld}|\cdot)$ following Algorithm 1
      Flip u_{ld} with probability $[p(u_{ld}|\cdot)^{-1}-1]^{-1}$
    end for
  end for
  Set λ to its MLE according to Eq. (5).
end for

---

rewrite the likelihood as $\sigma(\lambda)^P\sigma(-\lambda)^{ND-P}$ which can be subsequently maximised with respect to λ to yield the update

$$\sigma(\hat{\lambda}) = \frac{P}{ND} . \quad (5)$$

The alternation between sampling $(\boldsymbol{U}, \boldsymbol{Z})$ and updating the dispersion parameter is carried out until convergence; see Algorithm 2 for all steps of this procedure.

3.3. Dealing with Missing Data

We can handle unobserved data, by marginalising the likelihood over the missing observations. More precisely, if $\boldsymbol{X} = (\boldsymbol{X}_{\text{obs}}, \boldsymbol{X}_{\text{mis}})$ is the decomposition of the full matrix into the observed part $\boldsymbol{X}_{\text{obs}}$ and the missing part $\boldsymbol{X}_{\text{mis}}$, after marginalisation, the initial likelihood $p(\boldsymbol{X}|\boldsymbol{U}, \boldsymbol{Z}, \lambda)$ simplifies to $p(\boldsymbol{X}_{obs}|\boldsymbol{U}, \boldsymbol{Z}, \lambda)$. Then, a naïve implementation could be based on indexing the observed components inside matrix $\boldsymbol{X}$ and modifying the inference procedure so that the posterior conditionals of z_{nl} and u_{ld} involve only sums over observed elements. A simpler, equivalent implementation, which we follow in our experiments, is to represent the data as $\tilde{x}_{nd} \in \{-1, 0, 1\}$ where missing observations are encoded as zeros, each contributing the constant factor $\sigma(0) = 1/2$ to the full likelihood, so that $p(\boldsymbol{X}|\boldsymbol{U}, \boldsymbol{Z}, \lambda) = C\,p(\boldsymbol{X}_{obs}|\boldsymbol{U}, \boldsymbol{Z}, \lambda)$, where C is a constant. Thus, the missing values do not contribute to the posterior over $\boldsymbol{U}$ and $\boldsymbol{Z}$ which is also clear from the form of the full conditionals in Eq. (4) that depend on a sum weighted by x_{nd}s. For the update of the dispersion parameter in Eq. (5)), we need to subtract the number of all missing observations in the denominator. The dispersion now indicates the fraction of correct prediction in the observed data. Following this inference procedure, we can impute missing data based on a Monte Carlo estimate of
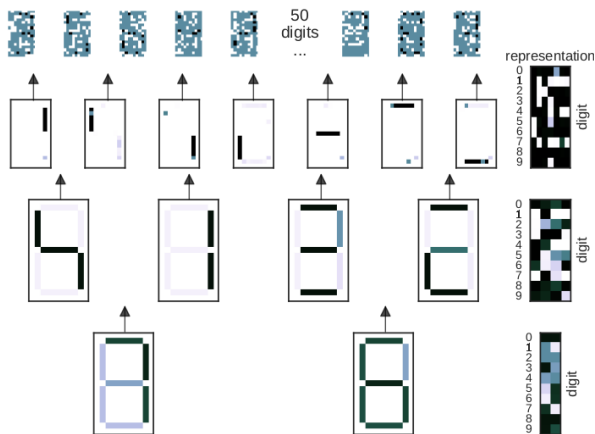
Figure 2. An OrMachine with 3 hidden layers is trained to reconstruct 50 calculator digits with 70% of observations missing. The rows depict increasingly abstract layers of the model. Shown are the latent prototypes fed forward to the data layer. Variables are arranged to 17×10 images for interpretation. The right sides show the corresponding posterior means for representations of the partially observed input digits.

the predictive distribution of some unobserved x_{nd} as

$$\frac{1}{S} \sum_{s=1}^{S} p(x_{nd}|\boldsymbol{U}^{(s)}, \boldsymbol{Z}^{(s)}, \hat{\lambda}) , \qquad (6)$$

where each $(\boldsymbol{U}^{(s)}, \boldsymbol{Z}^{(s)})$ is a posterior sample. A much faster approximation of the predictive distribution is obtained by $p(x_{nd}|\hat{\boldsymbol{U}}, \hat{\boldsymbol{Z}}, \hat{\lambda})$, where we simply plug the posterior mean estimates for $(\boldsymbol{U}, \boldsymbol{Z})$ into the predictive distribution. For the simulated data in Section 4.2, we find both methods to perform equally well and therefore follow the second, faster approach for all remaining experiments.

3.4. Multi-Layer OrMachine

BooMF learns patterns of correlation in the data. In analogy to multi-layer neural networks, we can build a hierarchy of correlations by applying another layer of factorisation to the factor matrix $\boldsymbol{Z}$. This is reminiscent of the idea of deep exponential families, as introduced by Ranganath et al. (2015). The ability to learn features at different levels of abstraction is commonly cited as an explanation for the success that deep neural networks have across many domains of application (Lin & Tegmark, 2016; Bengio et al., 2013). In the present setting, with stochasticity at every step of the generative process and posterior inference, we are able to infer meaningful and interpretable hierarchies of abstraction.

To give an example, we determine the optimal multi-layer architecture for representing the calculator digit toy dataset as introduced in Fig. 1. We observe 50 digits and consider 70% of the data points randomly as unobserved. We then train multi-layer OrMachines with various depths and layer widths, iterating through the individual layers during 200 iterations of burn-in. We then draw 200 samples from each consecutive layer with the remaining layers held fixed to their MAP estimate. In order to enforce distributed representations, we choose independent Bernoulli sparsity priors for the codes: $p(u_{ld}) = [0.01, 0.05, 0.2]$ for each layer, respectively. Superior performance in reconstructing the unobserved data is achieved by a 3-hidden layer architecture with hidden layers of size $L_1 = 7$, $L_2 = 4$, $L_3 = 2$. This 3-layer model reduces the reconstruction error from 1.4% to 0.4% compared to the single-layer model with width $L = 7$. Maximum likelihood estimates of the dispersion for the three layers are $\hat{\boldsymbol{\lambda}} = [1.0, 0.93, 0.8]$. The first layer infers the seven bars that compose all digits, the following layer infers dominant groupings of these bars, and so on. In Fig. 2, we plot the probabilities that each prototype induces in the observation layer. They are given by the means of the posterior predictive as described in the the previous section, conditioned on the one-hot activations of $\boldsymbol{z}_n$ with $z_{nl} \in \{0, 1\}$ and $\sum_l z_{nl} = 1$. Alongside, we depict the average posterior mean of the corresponding representations for each digit in the training data. This example illustrates that the multi-layer OrMachine infers interpretable higher-order correlations and is able to exploit them to achieve significant improvements in missing data imputation.

3.5. Practical Implementation and Speed

The algorithm is implemented in Python with the core sampling routines in compiled Cython. Code is avaialble on GitHub[1]. Binary data is represented as $\{-1, 1\}$ with missing data encoded as 0. This economical representation of data and variables as integer types simplifies computations considerably. Algorithm 1 is implemented in parallel across the observations $[n] = \{1, \ldots, N\}$ and conversely updates for u_{ld} are implemented in parallel across all features $[d] = \{1, \ldots, D\}$. The computation time scales linearly in each dimension. A single sweep through high-resolution calculator digits toy dataset with $ND = 1.7 \times 10^6$ data points and $L = 7$ latent dimensions takes approximately 1 second on a desktop computer. A single sweep through the approximately 1.4×10^{10} data points presented in the biological example in Section 5.2 with $L = 2$ latent dimensions takes approximately 5 minutes executed on 24 computing cores. For all examples presented here 100 iterations suffice for the algorithm to converge to a (local) posterior mode.

4. Experiments on Simulated Data

In this section, we probe the performance of the OrMachine (OrM) at random matrix factorisation and completion
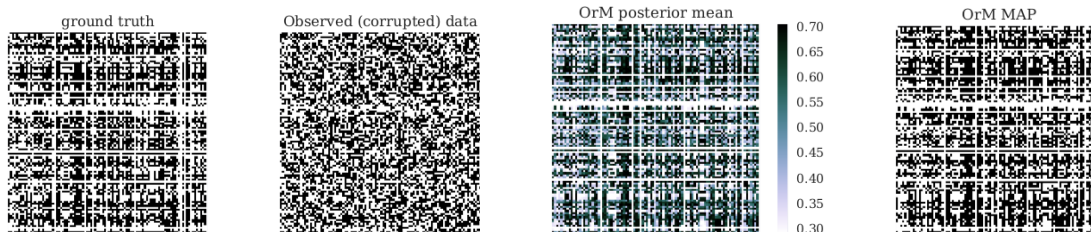

Figure 3. Illustration of the matrix factorisation task for a 100×100 matrix of rank 7. The posterior means estimate the probability of each data point to take a value of one. MAP estimates are computed by rounding to the closest integer.

tasks. Message passing (MP) has been shown to compare favourably with other state-of-the-art methods for BooMF that we introduced in Section 2 (Ravanbakhsh et al., 2016). It therefore is the focus of our comparison. The following settings for MP and the OrM are used throughout our experiments, unless mentioned otherwise. For MP, we use the Python implementation provided by the authors. We also proceed with their choice of hyper-parameters, as experimentation with different learning rates and maximum number of iterations did not lead to any improvements. For both methods, we set the priors $p(u)$ and $p(z)$ to the factor matrices' expected value based on the density of the product matrix in an Empirical-Bayes fashion. The only exception is MP in the matrix completion task, where uniform priors, as used by Ravanbakhsh et al. (2016), lead to slightly better performance. For the OrM, we initialise the parameters uniformly at random and draw 100 iterations after 100 samples of burn-in. Note that around 10–100 sampling steps are usually sufficient for convergence.

4.1. Random Matrix Factorisation

We generate a quadratic matrix $\boldsymbol{X} \in \{0,1\}^{N \times N}$ of rank L by taking the Boolean product of two random $N \times L$ factor matrices. The Boolean product X of two rank L binary matrices that are sampled i.i.d. from a Bernoulli distribution with parameter p has an expected value of $E(X) = 1 - (1 - p^2)^L$. Since we generally prefer X to be neither sparse nor dense, we fix its expected density to $1/2$, unless stated otherwise. This ensures that a simple bias toward zeroes or ones in either method is not met with reward. Bits in the data are flipped at random with probabilities ranging from 5% to 50%. Factor matrices of the correct underlying dimension are inferred and the data is reconstructed from the inferred factorisation. An example of the task is shown in Fig. 3.

Results for the reconstruction error, defined as the fraction of misclassified data points, are depicted in Fig. 4. All experiments were repeated 10 times with error bars denoting standard deviations. The OrM outperforms MP under all conditions, except when both methods infer equally error-free reconstructions. Fig. 4 (top) reproduces the experi-
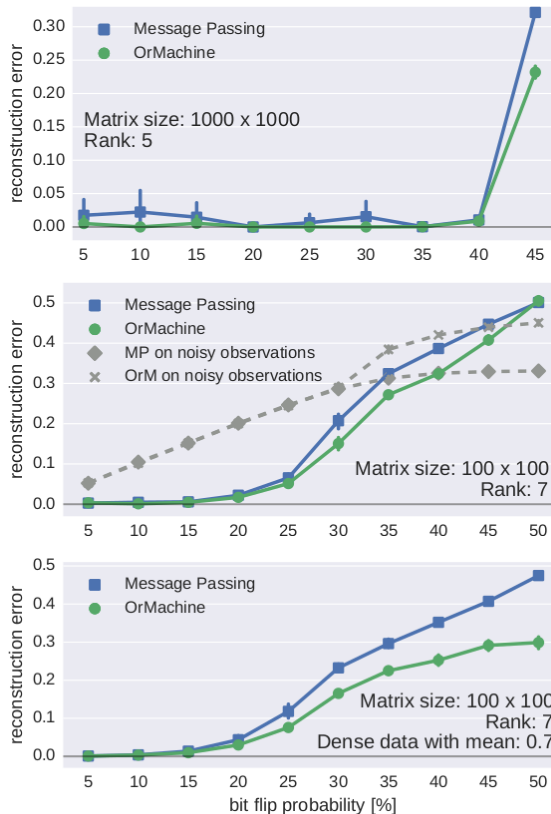

Figure 4. Comparison of OrMachine and message passing for BooMF for random matrices of different size, rank and density. Compare to Fig. 2 in Ravanbakhsh et al. (2016).

mental settings of Fig. 2 in Ravanbakhsh et al. (2016). We find that the OrMachine enables virtually perfect reconstruction of a 1000×1000 matrix of rank $L = 5$ for up to 35% bit flip probability. Notably, MP performs worse for smaller noise levels. It was hypothesised by Ravanbakhsh et al. (2016) that symmetry breaking at higher noise levels helps message passage to converge to a better solution. Fig. 4 (middle) demonstrates the consistently improved performance of the OrMachine for a more challenging example of 100×100 matrices of rank 7. The reconstruction performance of both methods is similar for lower noise levels, while the OrMachine consistently out-
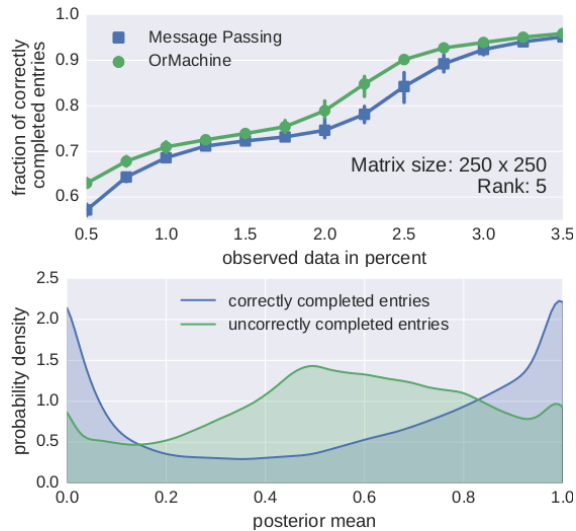
Figure 5. Matrix completion performance for simulated low rank matrices (top) and kernel density estimate of the distribution of posterior means for inferred matrix entries (bottom).

Table 1. Collaborative filtering performance for MovieLens 1M and 100k dataset. Given are the percentages of correctly reconstructed unobserved data as means from 10 random repetitions. Compare to Table 1 in Ravanbakhsh et al. (2016), who also provide comparison to other state-of the art methods. Their results for message passing were independently reproduced. The multi-layer OrMachine has two hidden layers of size 4 and 2, respectively.

| | OBSERVED PERCENT. OF AVAILABLE RATINGS | | | | | |
	1%	5%	10%	20%	50%	95%
100K						
ORM	58.5	63.5	64.9	66.4	68.9	70.0
MP	52.8	60.7	63.0	65.2	67.5	69.5
MULTI LAYER ORM	58.5	63.5	65.2	66.5	68.8	70.1
1M						
ORM	63.4	67.0	68.5	69.8	70.9	71.2
MP	56.7	64.9	67.2	68.8	70.7	71.5
MULTI LAYER ORM	63.8	67.2	68.6	70.0	71.4	72.1

performs MP for larger noise levels. For biased data with $\mathbb{E}[x_{nd}] = 0.7$ in Fig. 4 (bottom), we observe a similar pattern with a larger performance gap for higher noise levels. Even for a bit flip-probability of 50% the OrMachine retains a reconstruction error of approximately 30%, which is achieved by levering the bias in the data.

Fig. 4 (middle) also shows the reconstruction error on the observed data, indicating that MP overfits the data more than the OrM for larger noise levels. This may contribute to the improved performance of the OrMachine.

4.2. Random Matrix Completion

We further investigate the problem of matrix completion or collaborative filtering, where bits of the data matrix are unobserved and reconstructed from the inferred factor matrices. Following the procedure outlined in Section 4.1, we generate random matrices of rank 5 and size 250×250. We only observe a random subset of the data, ranging from 0.5% and 3.5%. The missing data is reconstructed from the inferred factor matrices. As shown in Fig. 5, the OrMachine outperforms message passing throughout. The plot indicates means and standard deviations from 10 repetitions of each experiment.

Notably, the OrMachine does not only provide a MAP estimate, but also an estimate of the posterior probability for each unobserved data point x_{nd}. Fig. 5 (bottom) shows an estimate of the density of the posterior means for the correctly and incorrectly completed data points. The distribution of incorrect predictions peaks around a probability of $1/2$, indicating that the OrMachine's uncertainty about

its reconstruction provides further useful information about the missing data. For instance, this information can be used to control for false positives or false negatives, simply by setting a threshold for the posterior mean.

5. Experiments on Real-World Data

5.1. MovieLens Matrix Completion

We investigate the OrMachine's performance for collaborative filtering on a real-world dataset. The MovieLens-1M dataset[2] contains 10^6 integer film ratings from 1 to 5 from 6000 users for 4000 films, i.e. 1/24 of the possible ratings are available. Similarly, the MovieLens 100k dataset contains 943 users and 1682 films. Following Ravanbakhsh et al. (2016), we binarise the data taking the global mean as threshold. We observe only a fraction of the available data, varying from 1% to 95%, and reconstruct the remaining available data following the procedure in Section 4.2 with $L = 2$ latent dimensions. Reconstruction accuracies are given as fractions of correctly reconstructed unobserved ratings in Table 1. The given values are means from 10 randomly initialised runs of each algorithm. The corresponding standard deviations are always smaller than 0.2%. The OrMachine is more accurate than message passing in all cases, except for the 1M dataset with 95% available ratings. The OrMachine's advantage is particularly significant if only little data is observed. Increasing the latent dimension L to values of 3 or 4 yields no consistent improvement, while a further increase is met with diminishing

---

[2]The MovieLens dataset is available online: `https://grouplens.org/datasets/movielens/`.
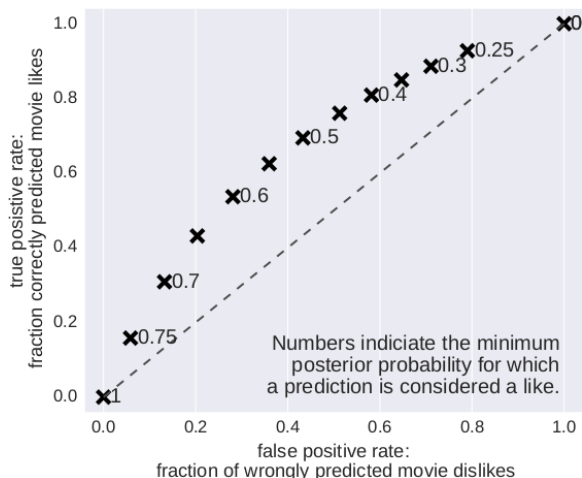
Figure 6. ROC curve for MovieLens 100k data, adjusting the threshold for when a prediction is considered a like. 10% of the available data were observed and used for inference with an OrM of size $L = 2$. Predictions were tested on the remaining 90%.
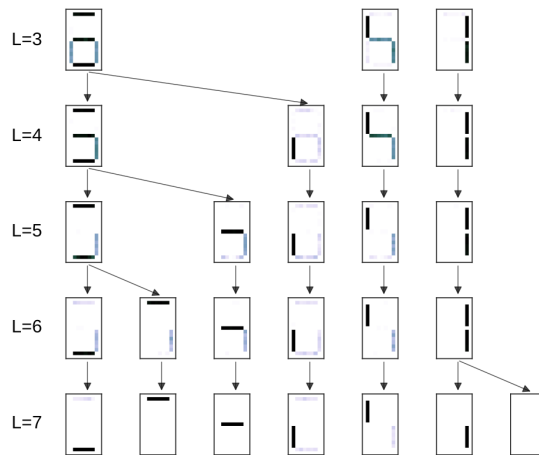

Figure 7. Hierarchy in the latent features of calculator digits. The arrows indicate the split of codes into more distributed codes of lower density. They are inferred as latent variables in an OrMachine, with codes from the model of size L as data and codes from model with of size $L+1$ as fixed codes.

returns. We achieve the best within-sample performance for a two-layer OrMachine with different architectures performing best for different amounts of observed data. An OrMachine with two hidden layers of sizes 4 and 2 respectively yields the best average performance. As indicated in Table 1, it provides better results throughout but exceeds the performance of the shallow OrMachine rarely by more than 1%. This indicates that there is not much higher order structure in the data, which is unsurprising given the sparsity of the observations and the low dimensionality of the first hidden layer.

We illustrate a further advantage of full posterior inference for collaborative filtering. We can choose a threshold for how likely we want a certain prediction to take a certain value and trade off false with true positives. A corresponding ROC curve for the MovieLens 100k dataset, where 10% of the available data was observed, is shown in Fig. 6.

5.2. Explorative Analysis of Single Cell Gene Expression Profiles

Single-cell RNA expression analysis is a revolutionary experimental technique that facilitates the measurement of gene expression on the level of a single cell (Blainey & Quake, 2014). In recent years this has led to the discovery of new cell types and to a better understanding of tissue heterogeneity (Trapnell, 2015). The latter is particularly relevant in cancer research where it helps to understand the cellular composition of a tumour and its relationship to disease progression and treatment (Patel et al., 2014). Here we apply the OrMachine to binarised gene expression profiles of about 1.3 million cells for about 28 thousand genes per cell. Cell specimens were obtained from cortex, hip-

pocampus and subventricular zone of E18 (embryonic day 18) mice; the data is publicly available[3]. Only 7% of the data points are non-zero. We set all non-zero expression levels to one, retaining the essential information of whether or not a particular gene is expressed. We remove genes that are expressed in fewer than 1% of cells with roughly 11 thousand genes remaining. This leaves us with approximately 1.4×10^{10} data points. We apply the OrMachine for latent dimensions $L = 2, \dots, 10$. The algorithm converges to a posterior mode after 10–20 iteration, taking roughly an hour on a 4-core desktop computer and 10–30 minutes on a cluster with 24 cores. We draw 125 samples and discard the first 25 as burn-in.

Factorisations with different latent dimensionality form hierarchies of representations, where features that appear together in codes for lower dimensions are progressively split apart when moving to a higher dimensional latent space. We illustrate this approach to analysing the inferred factorisations on calculator digits in Fig. 7. Each row corresponds to an independently trained OrMachine with the dimensionality L increasing from 3 to 7. We observe denser patterns dividing up consecutively until only the seven constituent bars remain. This is a form of hierarchical clustering that, in contrast to traditional methods, does not impose any hierarchical structure on the model. We perform the same analysis on the single cell gene expression data with the results for both, gene patterns and specimen patterns shown in Fig. 8. This Figure should be interpreted in analogy to Fig. 7. Furthermore, we run a gene set enrichment analysis for the genes that are unique to each inferred code, looking for associated biological states. This
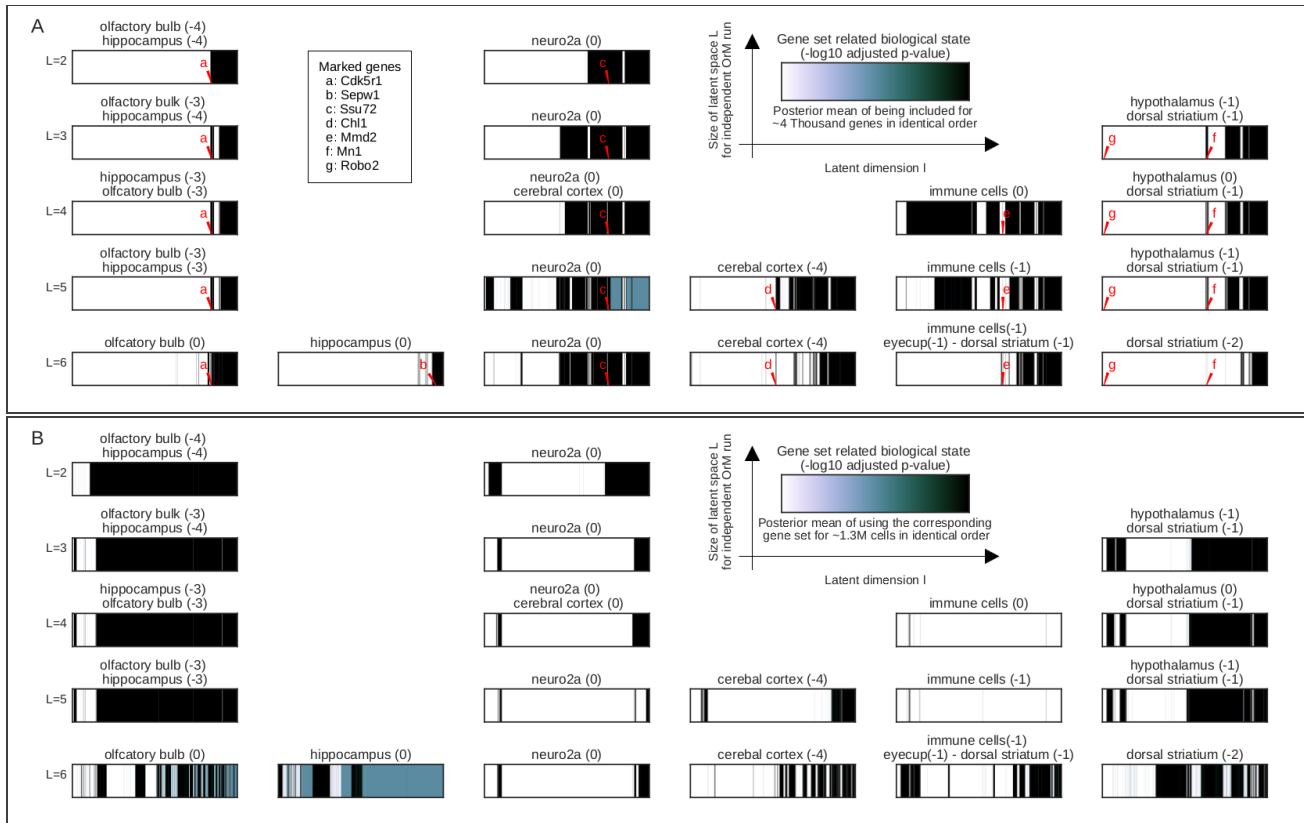
[3] https://support.10xgenomics.com

Figure 8. Hierarchy in the latent representations of genes (A) and specimens (B) under variation of the latent dimensionality. Rows with the same dimensionality ($L = 2 \ldots 6$) in the top and bottom box correspond to the same OrMachine factorisation. Rows with different dimensionality are trained independently. Importantly, the ordering of genes/specimens is identical for all subplots. Each subplot (A) describes set of expressed genes, limited to the approximately 4k out of 11k analysed genes that are used in at least one code. Subplots in (B) describe representations of cell specimens in terms of which of the gene sets they express. See legend in each box for more details.

is done using the Enrichr analysis tool (Chen et al., 2013) and a mouse gene atlas (Su et al., 2004). Biological states are denoted above each code, together with the logarithm to base 10 of their adjusted p-value. Increasing the latent dimensionality leads to a more distributed representation with subtler, biologically plausible patterns. The columns in Fig. 8 are ordered to emphasise the hierarchical structure. E.g., in the first column for $L = 5$ and second column for $L = 6$, a gene set with significant overlap to two biological processes (olfactory bulb and hippocampus) splits into two gene sets each corresponding to one of the two processes. In the assignment of cells to these sets (Fig. 8B), this is associated with an increase in posterior uncertainty as to which cell expresses this property. The significance levels of the associated biological processes drop from p-values on the order of 10^{-3} to p-values on the order of 1. In addition, typical genes for each of the biological states are annotated (Lopez-Bendito et al., 2007; Zheng et al., 2008; Demyanenko et al., 2010; Upadhya et al., 2011; Raman et al., 2013). This examples illustrates the OrMachine's ability to scale posterior inference to massive datasets. It enables the

discovery of readily interpretable patterns, representations and hierarchies, all of which are biologically plausible.

6. Conclusion

We have developed the OrMachine, a probabilistic model for Boolean matrix factorisation. The extremely efficient Metropolised Gibbs sampler outperforms state-of-the-art methods in matrix factorisation and completion. It is the first method that infers posterior distributions for Boolean matrix factorisation, a property which is highly relevant in practical applications where full uncertainty quantification matters. Despite full posterior inference, the proposed method scales to very large datasets. We have shown that tens of billions of data points can be handled on commodity hardware. The OrMachine can readily accommodate missing data and prior knowledge. Layers of OrMachines can be stacked, akin to deep belief networks, inferring representations at different levels of abstraction. This leads to improved reconstruction performance on simulated and real world data.

References

Agrawal, Rakesh, Srikant, Ramakrishnan, et al. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, volume 1215, pp. 487–499, 1994.

Bengio, Y., Courville, A., and Vincent, P. Representation learning: a review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35 (8):1798–1828, 2013. doi: 10.1109/tpami.2013.50.

Blainey, Paul C. and Quake, Stephen R. Dissecting genomic diversity, one cell at a time. *Nat. Methods*, 11(1): 19–21, Jan 2014.

Blei, David M. Probabilistic topic models. *Communications of the ACM*, 55(4):77, 2012. doi: 10.1145/ 2133806.2133826.

Chen, Edward Y, Tan, Christopher M, Kou, Yan, Duan, Qiaonan, Wang, Zichen, Meirelles, Gabriela, Clark, Neil R, and Ma'ayan, Avi. Enrichr: Interactive and collaborative html5 gene list enrichment analysis tool. *BMC Bioinformatics*, 14(1):128, 2013. doi: 10.1186/ 1471-2105-14-128.

Demyanenko, G. P., Siesser, P. F., Wright, A. G., Brennaman, L. H., Bartsch, U., Schachner, M., and Maness, P. F. L1 and chl1 cooperate in thalamocortical axon targeting. *Cerebral Cortex*, 21(2):401–412, 2010. doi: 10.1093/cercor/bhq115.

Lázaro-Gredilla, Miguel, Liu, Yi, Phoenix, D Scott, and George, Dileep. Hierarchical compositional feature learning. *arXiv preprint arXiv:1611.02252*, 2016.

Lin, Henry W. and Tegmark, Max. Why does deep and cheap learning work so well? *arXiv preprint arXiv:1608.08225*, 2016.

Liu, J. Miscellanea. peskun's theorem and a modified discrete-state gibbs sampler. *Biometrika*, 83(3):681– 682, 1996. doi: 10.1093/biomet/83.3.681.

Lopez-Bendito, G., Flames, N., Ma, L., Fouquet, C., Meglio, T. Di, Chedotal, A., Tessier-Lavigne, M., and Marin, O. Robo1 and robo2 cooperate to control the guidance of major axonal tracts in the mammalian forebrain. *Journal of Neuroscience*, 27(13):3395–3407, 2007. doi: 10.1523/jneurosci.4605-06.2007.

Meeds, Edward, Ghahramani, Zoubin, Neal, Radford M, and Roweis, Sam T. Modeling dyadic data with binary latent factors. *Advances in neural information processing systems*, 19:977, 2007.

Miettinen, Pauli and Vreeken, Jilles. Mdl4bmf: Minimum description length for boolean matrix factorization. *ACM Trans. Knowl. Discov. Data*, 8(4):18:1–18:31, October 2014. ISSN 1556-4681. doi: 10.1145/2601437.

Miettinen, Pauli, Mielikäinen, Taneli, Gionis, Aristides, Das, Gautam, and Mannila, Heikki. *The Discrete Basis Problem*, pp. 335–346. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-46048-0. doi: 10.1007/ 11871637_33.

Patel, A. P., Tirosh, I., Trombetta, J. J., Shalek, A. K., Gillespie, S. M., Wakimoto, H., Cahill, D. P., Nahed, B. V., Curry, W. T., Martuza, R. L., Louis, D. N., Rozenblatt-Rosen, O., Suva, M. L., Regev, A., and Bernstein, B. E. Single-cell rna-seq highlights intratumoral heterogeneity in primary glioblastoma. *Science*, 344(6190):1396– 1401, 2014. doi: 10.1126/science.1254257.

Peskun, Peter H. Optimum monte-carlo sampling using markov chains. *Biometrika*, 60(3):607–612, 1973.

Raman, Arjun V., Pitts, Matthew W., Seyedali, Ali, Hashimoto, Ann C., Bellinger, Frederick P., and Berry, Marla J. Selenoprotein w expression and regulation in mouse brain and neurons. *Brain and Behavior*, 3(5): 562–574, 2013. doi: 10.1002/brb3.159.

Ranganath, Rajesh, Tang, Linpeng, Charlin, Laurent, and Blei, David M. Deep exponential families. In *AISTATS*, 2015.

Ravanbakhsh, Siamak, Póczos, Barnabás, and Greiner, Russell. Boolean matrix factorization and noisy completion via message passing. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *JMLR: W&CP*, 2016.

Streich, Andreas P., Frank, Mario, Basin, David, and Buhmann, Joachim M. Multi-assignment clustering for boolean data. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, - 2009. doi: 10.1145/1553374.1553498.

Su, A. I., Wiltshire, T., Batalov, S., Lapp, H., Ching, K. A., Block, D., Zhang, J., Soden, R., Hayakawa, M., Kreiman, G., Cooke, M. P., Walker, J. R., and Hogenesch, J. B. A gene atlas of the mouse and human protein-encoding transcriptomes. *Proceedings of the National Academy of Sciences*, 101(16):6062–6067, 2004. doi: 10.1073/pnas.0400782101.

Su, Xiaoyuan and Khoshgoftaar, Taghi M. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2, January 2009. ISSN 1687-7470. doi: 10. 1155/2009/421425.

Trapnell, Cole. Defining cell types and states with single-cell genomics. *Genome Research*, 25(10):1491–1498, 2015. doi: 10.1101/gr.190595.115.

Upadhya, Sudarshan C., Smith, Thuy K., Brennan, Peter A., Mychaleckyj, Josyf C., and Hegde, Ashok N. Expression profiling reveals differential gene induction underlying specific and non-specific memory for pheromones in mice. *Neurochemistry International*, 59 (6):787–803, 2011. doi: 10.1016/j.neuint.2011.08.009.

Wood, Frank, Griffiths, Thomas, and Ghahramani, Zoubin. A non-parametric bayesian method for inferring hidden causes. *arXiv preprint arXiv:1206.6865*, 2012.

Zheng, Yue, Cheng, Xiao-Rui, Zhou, Wen-Xia, and Zhang, Yong-Xiang. Gene expression patterns of hippocampus and cerebral cortex of senescence-accelerated mouse treated with huang-lian-jie-du decoction. *Neuroscience Letters*, 439(2):119–124, 2008. doi: 10.1016/j.neulet. 2008.04.009.