Matrix Factorization for Learning Metagenomic Pathways and Species

Silja Polvi-Huttunen

December 8, 2014

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta/Osasto — Fakultet/Sektion — Faculty Faculty of Science	Laitos — Institution — Department Department of Mathematics and Statistics

Tekijä — Författare — Author
Silja Polvi-Huttunen

Työn nimi — Arbetets titel — Title

Matrix Factorization for Learning Metagenomic Pathways and Species

Oppiaine — Läroämne — Subject

Applied mathematics

Työn laji — Arbetets art — Level Master's thesis	Aika — Datum — Month and year December 2014	Sivumäärä — Sidoantal — Number of pages 75 p

Tiivistelmä — Referat — Abstract

This work considers learning meaningful sets of chemical reactions called pathways and groups of species called Operational Taxonomical Units (OTUs) from metagenomic data. The methods are based on Nonnegative Matrix Factorization (NMF).

The rows of our data matrix correspond to metagenomic samples and columns correspond to chemical reactions present in the samples. In order to learn both pathways and OTUs as well as relationships between them, we consider ways to factorize the data matrix into three factors instead of two. Denoting the *samples* $\times$ *reactions* data matrix by V, our factorization problem setting is to find nonnegative matrices W, H and P so that $V \approx WHP$. The matrix W tells what OTUs are present in each of the samples, P defines pathways as combinations of reactions while H describes what pathways are implemented by which OTUs.

We first discuss two standard NMF algorithms based on different objective functions and four sparsity constrained variants. Sparsity constrained variants are designed to produce output matrices with few values significantly above zero. We are interested in sparser variants because metagenomic pathways are short, thus the method should find a representation where only a small set of reactions is present in each pathway.

We describe how using a standard two-factor NMF method twice yields a three-factor representation. We briefly mention an existing method, Nonnegative Matrix Tri-factorization (NMTF), that learns all three matrices W, H and P simultaneously. However, this method applies hard orthogonality constraints, *i.e.* it only finds solutions where the matrices W and P are orthogonal. Because of this constraint, NMTF is not suitable in our biological problem setting. We introduce an unconstrained method called NMF3 as well as a sparsity constrained variant SNMF3 based on Sparse Nonnegative Matrix Factorization (SNMF) and show how both of these algorithms can be derived.

In order to compare the different algorithms' performance, we have built two synthetic data sets. Both sets are based on human intestinal species and pathway information available in an existing biological database. One of the data matrices can be exactly factorized into the underlying matrices used to generate the data. The other data set is built through simulating a sampling process that introduces noise and strictly limits the number of observed reactions per sample.

We tested factorization methods discussed in the thesis on both data sets, using 100 to 1500 samples. We compare the methods and show and discuss the results. We found differences between NMF variants that use different objective functions. Many methods perform well on our task, surprisingly even in the case where the number of pathways is greater than the number of samples. Varying the number of samples affected the results less than we expected. Instead, we found that all algorithms performed significantly better on the factorizable data than on the simulated set. We conclude that the number of available metagenomic samples does not dramatically affect the performance of the factorization methods. More important is the quality of the samples.

Avainsanat — Nyckelord — Keywords
Nonnegative Matrix Factorization, metagenomics

Säilytyspaikka — Förvaringsställe — Where deposited
Kumpula library

Muita tietoja — Övriga uppgifter — Additional information

Contents

1 Introduction

In this work, we aim to learn meaningful knowledge from raw metagenomic samples. A metagenomic sample contains genetic information from many organisms present in an environment, such as sea water, soil, or bacteria from human intestine [5, 6, 7]. Some of this information codes enzymes that catalyze chemical reactions. By comparing the metagenomic sample data to known enzyme coding sequences in an existing database, it is possible to predict what reactions might happen in the organisms in the environment.

Our goal is both to group these reactions into meaningful processes, called *pathways*, and to find out what kind of organisms are responsible for each pathway. An *operational taxonomical unit*, *OTU*, is a group of species that are responsible for similar functions, so they implement many of the same pathways.

Both identifying species and predicting pathways in metagenomic samples are known problems. A common approach [17, 19, 21] is to compare [15, 16] the genetic data in the sample against known organisms' genomes or known pathways in existing databases [1, 2, 3, 4]. These methods are not well suited for identifying species that lack a reference genome in the database, such as previously unknown species. Unsupervised machine learning methods, such as binning [22, 23, 24], use statistical methods to group genetic data in samples into OTU's.

To the best of our knowledge, this is the first attempt to extract both OTUs and pathways as well as relationships between them.

In this thesis, we tackle the indentification problem with Nonnegative Matrix Factorization (NMF) [33, 34, 35], a popular unsupervised machine learning method capable of handling thousands of samples at a time, effectively using information from different and also diverse samples. Given a data matrix V, NMF finds two nonnegative matrices W and B, whose matrix product approximates the data: $V \approx WB$. In other words, the data is represented as linear combinations of t_1 base vectors, rows in B. t_1 is called the *inner dimension* of the factorization, and the dimension of the data is reduced to t_1. The base vectors represent some latent features of the data, and t_1 is chosen according to the number of the latent features we want to extract.

In order to reach both our goals, we further want to find a base for the base vectors, which means factorizing the base matrix B further into two matrices, $B \approx HP$. Together these factorizations give a factorization of the data matrix into three matrices, $V \approx WHP$.

The input data is organized in a matrix so that each row corresponds to a metagenomic sample and each column to a reaction. Each element (i, j) indicates the amount of evidence of the reaction j found in sample i. When this sample-reaction matrix V is factorized into three matrices, $V \approx WHP$, each column in the matrix W corresponds to a predicted OTU and each row in the matrix P to a pathway. Thus, W shows which OTUs are present in each sample, P defines the learned pathways as combinations of reactions, and

H indicates what pathways each OTU implements.

We discuss some existing NMF algorithms and show how they can be used to factorize the data matrix into three factors by factorizing twice into two matrices, as shown above. In addition to standard unconstrained techniques, we investigate some sparsity constrained variants. We briefly mention an existing orthogonality constrained technique, Nonnegative Matrix Tri-Factorization (NMTF) [40], that learns all three factor matrices simultaneously, and derive our own unconstrained and sparsity constrained variants.

We choose some NMF-based methods and test them on the artificial and simulated data sets introduced in Section 7.

The rest of this thesis is organized as follows. In Section 2, we introduce notation. In Section 3 we discuss some NMF methods that are currently available, and Section 4 discusses factorizing a matrix into three matrices using these methods. In Sections 5, we discuss an existing three-matrix factorization technique and in Section 6 we derive our own variants. Artificial and simulated data used to test the algorithms are introduced in Section 7, and results are shown in Section 8. Source code of all implementations we used are available on the Internet (github.com/stpolvi/matrix-factorization-algorithms).

1.1 Metagenomics

Traditionally, microbial samples are cultivated in a laboratory. Because most microbial species cannot be cultivated, traditional methods do not allow studying these species, and our view of microbial life is skewed [8, 9]. In metagenomics, genome sequences are extracted from uncultivated samples [10, 11, 12]. Each sequence read is randomly extracted from any individual organism present in the sample. Many reads come from dominant species, while some species may not be represented in the data at all, depending on the sequencing depth [13].

Metagenomics allows discovering new species in very diverse environments and studying evolution, genetic patterns and functional potential of a sample as a community rather than considering each species individually [14].

The combined amount of genetic information in a sample is vast. Advanced machine learning techniques are needed in the analysis, and it is computationally demanding.

1.2 Background

Methods such as binning [22, 23, 24] have been used to assign genetic data into groups that correspond to OTUs. The advantage of some of these methods is that they can analyze samples that contain OTUs whose reference genomes are not available.

Different definitions for pathways, such as *extreme pathways* [25] and *elementary flux modes* [26] have been suggested, and pathways have been examined using *e.g.* logic programming [27], graph-based methods [28, 29], heuristic search on enzymes whose metabolites are known [30] and convex analysis [25]. In this thesis, we simply define pathways as meaningful groups of enzymes. Our synthetic data sets are based on the MetaCyc database [3], and all validation is based on pathways available in MetaCyc.

Nonnegative Matrix Factorization has been used to analyze metagenomic data *e.g.* to find canonical sample types and functional profiles from marine environments. In 2012, Jiang *et al* [20] used 39 samples from 7 different kinds of marine or animal environments. The data matrix is (558×39), and each row corresponds to a pathway assigned to some of the samples. Their method found that 3 was a good inner dimension for the factorization, meaning that they only find three latent features, canonical sample types, from the data.

In 2012 as well, Jiang *et al* [21] used NMF to analyze 45 ocean samples from the Global Ocean Sampling project [5] and associated 8214 Protein Families (Pfams) [4]. They factorized this (8214×45) matrix into two matrices using inner dimension 5 to reveal five latent features, functional profiles.

One difference between these studies and ours is that we decompose our data matrix into three factors instead of one to reveal two sets of latent variables, OTUs and pathways, as well as connections between them. Another difference is the dimensions of the data matrix and the factorization. In the above-mentioned studies the number of extracted latent features was as small as 7.7 or 11 per cent of the smaller dimension of the data matrix. Our synthetic data sets contain 100, 500 or 1500 samples and at most 1234 associated reactions. The inner dimensions are 100 and 422: near the smaller dimension of the data or even bigger. The third difference is the type of the latent features. Whereas the previous studies found features that each represent one environment or sample type, we are interested in much more specific details, OTUs and pathways.

2 Notation

In this thesis, we denote matrices with uppercase letters, scalars with lowercase letters and vectors with boldface lowercase letters. For example, X would be a matrix, x a scalar and $\mathbf{x}$ a vector.

Notation	Meaning
$\frac{\partial}{\partial x} f$	the partial derivative of f with respect to x
$\sum\limits_{P(i)} a_i$	Sum over scalars a_i, where i goes through natural numbers that have the property P; for example $\sum\limits_{i \leq n} a_i = a_1 + a_2 + a_3 + \cdots + a_n$. If there are no natural numbers that have the property P, the sum is defined to be 0.
$V \in \mathbb{R}_{\geq 0}^{(n \times m)}$	V is a matrix of nonnegative real numbers, n rows and m columns
V_{ij}	the element on row i and column j of a matrix V
WH	matrix product of W and H; only defined when the number of columns in W is the same as the number of rows in H, say t, by $(WH)_{ij} = \sum\limits_{k \leq t} W_{ik} \cdot H_{kj}$ Note that we always use the dot $(\cdot)$ to denote scalar product, but never the matrix product.
W^T	matrix transpose of W, defined by $W^T_{ji} = W_{ij}$

The identity matrix I is defined by $I_{ii} = 1$ and $I_{ij} = 0$ when $i \neq j$. It is always square, and its size is clear from the context if not explicitly stated. Note that $AI = A$ and $IA = A$ hold for any matrix A.

Notation	Meaning
$\ln(x)$	natural logarithm of x
$\|x\|$	absolute value of x
$L^1(\mathbf{x})$	L^1 norm of a vector $\mathbf{x} = (x_1, x_2, ..., x_m)$, defined as the sum over the absolute values of elements in $\mathbf{x}$: $L^1(\mathbf{x}) = \sum_{i\leq m} \|x_i\|$
$L^2(\mathbf{x})$	L^2 norm of a vector $\mathbf{x} = (x_1, x_2, ..., x_m)$, defined by $L^2(\mathbf{x}) = \sqrt{\sum_{i\leq m} x_i^2}$
$b(A\|B)$	Absolute error between A and B. Defined when both A and B are $(n \times m)$ matrices by $b(A\|B) = \sum_{i\leq n} \sum_{j\leq m} \|A_{ij} - B_{ij}\|$.
$e(A\|B)$	Euclidean squared error between A and B. Defined when both A and B are $(n \times m)$ matrices by $e(A\|B) = \sum_{i\leq n} \sum_{j\leq m} (A_{ij} - B_{ij})^2$.
$d(A\|B)$	The extended Kullback-Leibler divergence from A to B; typically, B is an estimate of A. Defined when both A and B are $(n \times m)$ matrices by $d(A\|B) = \sum_{i\leq n} \sum_{j\leq m} \left(A_{ij} \cdot \ln \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij} \right)$. It is required that the estimate B may only be zero where the true distribution A is zero. In case for some i and j, $A_{ij} = B_{ij} = 0$, we set the corresponding term $A_{ij} \cdot \ln \frac{A_{ij}}{B_{ij}}$ to equal zero.

$$\underset{\text{sample} \times \text{reaction}}{\begin{bmatrix} 0.3 & 0.2 & 0.2 & 0 & 0.2 & 0.1 \\ 0.3 & 0.1 & 0 & 0.2 & 0.4 & 0 \\ 0.4 & 0.2 & 0.2 & 0 & 0.1 & 0.1 \\ 0.2 & 0.2 & 0.2 & 0 & 0.1 & 0.3 \end{bmatrix}} \approx \underset{\text{sample} \times \text{OTU}}{\begin{bmatrix} 0.2 & 0.7 & 0.1 \\ 0.9 & 0.1 & 0 \\ 0 & 1 & 0 \\ 0 & 0.6 & 0.4 \end{bmatrix}} \underset{\text{OTU} \times \text{reaction}}{\begin{bmatrix} 0.3 & 0.1 & 0 & 0.2 & 0.4 & 0 \\ 0.4 & 0.2 & 0.2 & 0 & 0.1 & 0.1 \\ 0 & 0.1 & 0.2 & 0 & 0.2 & 0.5 \end{bmatrix}}$$

Figure 1: Factorization of sample-reaction matrix V into sample-OTU and OTU-reaction matrices W and H

3 Previously Available NMF Methods

Nonnegative Matrix Factorization (NMF) [33, 34] is a popular dimension reduction and clustering method that approximately factorizes a given data matrix into two nonnegative matrices.

3.1 Example

Assume we have n metagenomic samples and genetic evidence of m different chemical reactions happening in those samples collectively. We arrange the samples in a matrix $V \in \mathbb{R}_{\geq 0}^{(n \times m)}$, whose rows correspond to the samples and columns correspond to the reactions. The entry V_{ij} of this matrix is the amount of evidence of reaction j in sample i.

We can now present this sample-reaction matrix as a matrix product of two matrices, sample-OTU matrix W and OTU-reaction matrix H, as shown in Figure 1. This factorization explains the data so that each OTU is responsible for a set of reactions, and the reactions that happen in each sample is explained by linear combinations of OTUs in that sample.

This example motivates the nonnegativity constraint. For example, we interpret that entries of the (sample $\times$ OTU) matrix show how much evidence of each OTU there is in each sample, and the rows in the (OTU $\times$ reaction) matrix define distributions of reactions over each OTU. We do not allow a negative amount of genetic evidence or negative values to occur in distributions. Negative entries would also allow values to cancel out each other from the linear combinations when the matrices are multiplied.

3.2 Cost functions

Mathematically, the NMF problem setting is: Given $V \in \mathbb{R}_{\geq 0}^{(n \times m)}$ and $t \in \mathbb{N}$, find matrices $W \in \mathbb{R}_{\geq 0}^{(n \times t)}$ and $H \in \mathbb{R}_{\geq 0}^{(t \times m)}$ that minimize the reconstruction cost between the data V and the product WH. Different applications use different cost functions to measure the reconstruction cost, and Euclidean squared error and extended Kullback-Leibler (KL) divergence are two of the most popular.

Euclidean squared error between vectors is the square of their Euclidean distance, regarding the vectors as points in the Euclidean space. For $(n \times m)$ matrices V and $\hat{V}$, the squared error $e(V|\hat{V})$ is defined as the sum of squared distances between their rows, or as if they were $n \cdot m$-dimensional vectors,

$$e(V|\hat{V}) = \sum_{i \leq n} \sum_{j \leq m} (V_{ij} - \hat{V}_{ij})^2.$$

As a sum of squares, the value of this function is zero when all terms $V_{ij} - \hat{V}_{ij}$ are zero, which means V_{ij} is equal to $\hat{V}_{ij}$, and otherwise it is positive.

Standard KL divergence regards two vectors as a probability distribution and its estimate, and it measures how much information is lost if instead of the true distribution the estimate is used. It assumes the sum of entries in both vectors is one, as with distributions. It also assumes that the estimate never has the value zero in an index where the original distribution is nonzero. For an $(n \times m)$ matrix V and its estimate $\hat{V}$, whose rows do not necessarily sum to one, the *extended Kullback-Leibler divergence* is defined as

$$d(V|\hat{V}) = \sum_{i \leq n} \sum_{j \leq m} \left(V_{ij} \cdot \ln \left(\frac{V_{ij}}{\hat{V}_{ij}} \right) - V_{ij} + \hat{V}_{ij} \right).$$

In this equation, we set $V_{ij} \cdot \ln \left(\frac{V_{ij}}{\hat{V}_{ij}} \right)$ to zero in the case where both V_{ij} and $\hat{V}_{ij}$ are zero. As with the standard KL divergence, we assume $\hat{V}_{ij}$ is never zero when V_{ij} is nonzero because the value of the expression would grow to infinity.

3.3 Additive and multiplicative update rules

The nonnegative matrix factorization problem is in general NP-hard [32]. Exact polynomial time algorithms exist, if additional assumptions on the input matrix hold [31]. However, most NMF implementations do not find the global optimum of the cost function, and algorithms that find only a local optimum are widely used and sufficient to produce interesting results [34].

All NMF implementations discussed in this thesis are based on the general algorithm described in Algorithm 1. It randomly initializes the matrices W and H and iteratively updates one of them at a time, keeping the other one fixed. One of the matrices is also normalized in each iteration.

In update steps of Algorithm 1, W and H are updated with certain *update rules* that

```
**algorithm** ***generalNMF***($V \in \mathbb{R}_{\geq 0}^{(n \times m)}$, $t \in \mathbb{N}$)
    randomly initialize $W \in (0,1)^{(n \times t)}$ and $H \in (0,1)^{(t \times m)}$
    normalize $H$ row-wise
    **until** *convergence* **do**
        update $W$
        update $H$
        normalize $H$ row-wise
    **end until**
**end algorithm**
```

Algorithm 1: General NMF algorithm. Some variants use different normalization steps instead of normalizing H row-wise.

change each element of the matrices. The gradient descent method gives *additive update rules* that move the matrices towards the negative gradient of the cost function c by adding small numbers to its elements. More precisely, the rules update the elements by

$$W_{ab} := W_{ab} - \alpha_{ab} \cdot \frac{\partial}{\partial W_{ab}} c(V|WH)$$

and

$$H_{bc} := H_{bc} - \beta_{bc} \cdot \frac{\partial}{\partial H_{bc}} c(V|WH),$$

where $\frac{\partial}{\partial W_{ab}} c(V|WH)$ is the partial derivative of the cost function c with respect to the element W_{ab}, $\frac{\partial}{\partial H_{bc}} c(V|WH)$ is the partial derivative of the cost function c with respect to the element H_{bc} and α_{ab} and β_{bc} are carefully chosen, small step sizes. The disadvantage of additive update rules is that too small step sizes make the algorithm converge very slowly, while taking too large steps might move the updated matrix past the local optimum and at worst increase the reconstruction cost.

Multiplicative update rules bypass the step size selection problem and update the matrix elements by multiplication instead of addition. The multiplicative update rules that our NMF implementations use result from using a suitable function of the matrices W and H as the step size of additive rules. They were introduced by Lee and Seung [35] who characterized the multiplicative algorithms as “diagonally rescaled gradient descent, where the scaling factor is optimally chosen to ensure convergence”.

3.4 Normalizing constraints

Most common NMF implementations add a normalizing constraint to one of the matrices.

If $C \in \mathbb{R}^{t \times t}$ is an invertible matrix, *i.e.* there exists a matrix C^{-1} so that CC^{-1} is the identity matrix, it holds that

$$WH = WCC^{-1}H.$$

If D is a diagonal matrix whose diagonal elements are d_k, its inverse is the diagonal matrix whose diagonal elements are $\frac{1}{d_k}$ *i.e.* the reciprocals of d_k. Taking the product WD only scales each column of W by the corresponding diagonal element of D, and taking the product $D^{-1}H$ scales each row of H by the corresponding diagonal element of D^{-1}. If we select each diagonal element of D to be the sum over the corresponding row in H, setting

$$W := WD \text{ and } H := D^{-1}H$$

scales the rows of H to have unit L^1 norm without changing the reconstructed matrix WH. This makes convergence analysis of the algorithms easier compared to the approach where only H is rescaled and W is not changed.

This shows that while update rules minimize the reconstruction error, the scale of the matrices is not determined. For basic NMF algorithms whose objective functions only concern the product WH and not the matrices W and H individually, we can select any scale for the colunms of W or rows of H.

Most algorithms considered in this thesis set the L^1 norm of each row or each column of a matrix to 1, as shown in the example above. This is convenient for distributions. An exception to this is Nonnegative Matrix Factorization with sparseness constraints [36] discussed in Section 3.6.2. Its normalizing constraint sets the L^2 norm to unity instead of L^1 norm. This is implemented similarly as above, by selecting each diagonal element of D to equal the L^2 norm of the corresponding row of H.

In total, this thesis considers six previously available NMF variants. The objective functions of three of the algorithms are based on Kullback-Leibler divergence, and the others minimize Euclidean squared error. Some variants are designed to make one or both of the resulting matrices sparser.

3.5 Standard Nonnegative Matrix Factorization

In this section, we show how basic nonnegative matrix factorization algorithms can be derived. We consider two variants with different cost functions, squared error and KL divergence, and denote these variants by NMF-se and NMF-kl. Additive rules are based on the gradient descent method. Lee and Seung [35] give convergence proofs for the multiplicative update rules shown in Tables 1 and 2. We use their remarks on the relationship

$$H_{bc} := H_{bc} \cdot \frac{\left(W^T V\right)_{bc}}{(W^T W H)_{bc}}$$

$$W_{ab} := W_{ab} \cdot \frac{(V H^T)_{ab}}{(W H H^T)_{ab}}$$

Table 1: NMF-se update rules

$$H_{bc} := H_{bc} \cdot \frac{\displaystyle\sum_{i \le n} W^T_{bi} \cdot \frac{V_{ic}}{(WH)_{ic}}}{\displaystyle\sum_{i \le n} W^T_{bi}}$$

$$W_{ab} := W_{ab} \cdot \frac{\displaystyle\sum_{j \le m} \frac{V_{aj}}{(WH)_{aj}} \cdot H^T_{jb}}{\displaystyle\sum_{j \le m} H^T_{jb}}$$

Table 2: NMF-kl update rules

between additive and multiplicative rules to derive multiplicative rules.

Let $V \in \mathbb{R}^{(n \times m)}$ be a given data matrix, $c : \mathbb{R}^{(n \times m)} \times \mathbb{R}^{(n \times m)} \to \mathbb{R}$ a differentiable cost function, and t the desired inner dimension. The goal is to find an approximate factorization of V into two matrices, $V \approx WH$, where $W \in \mathbb{R}^{(n \times t)}$, $H \in \mathbb{R}^{(t \times m)}$, and each element of both W and H is nonnegative. We want to find W and H so that $c(V|WH)$ is locally minimized. We choose a scale for the base vectors and add a corresponding normalizing constraint, in this case that the rows in H must sum up to 1. This constrained optimization problem is solved by initializing the matrices W and H randomly, and iteratively updating them, in each step projecting H back to the normalization constraint set as described in Algorithm 1.

In the following, we assume both matrices W and H are initialized with nonnegative real values, the data matrix is a nonnegative $(n \times m)$ matrix, and the required inner dimension is t.

3.5.1 NMF update rules minimizing Euclidean squared error (NMF-se)

In this section, we consider the squared error cost function

$$e(V|WH) = \sum_{i \le n} \sum_{j \le m} \left(V_{ij} - (WH)_{ij}\right)^2 .$$

We first show details for the update rule for the matrix W.

To derive the update rule for W, we calculate the derivative of the cost with respect to W_{ab}. We start by calculating the partial derivative of $(WH)_{ij}$:

$$\begin{aligned}
\frac{\partial}{\partial W_{ab}}(WH)_{ij} &= \frac{\partial}{\partial W_{ab}} \sum_{k \le t} W_{ik} \cdot H_{kj} \\
&= \sum_{k<b} \frac{\partial}{\partial W_{ab}} W_{ik} \cdot H_{kj} + \sum_{k=b} \frac{\partial}{\partial W_{ab}} W_{ik} \cdot H_{kj} + \sum_{b<k\le t} \frac{\partial}{\partial W_{ab}} W_{ik} \cdot H_{kj} \\
&= \frac{\partial}{\partial W_{ab}} W_{ib} \cdot H_{bj} \\
&= \begin{cases} H_{bj} & \text{if } i = a, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}$$

Then, we calculate the derivative of the cost,

$$\begin{aligned}
\frac{\partial}{\partial W_{ab}} e(V|WH) &= \frac{\partial}{\partial W_{ab}} \sum_{i\leq n}\sum_{j\leq m} (V_{ij} - (WH)_{ij})^2 \\
&= 2 \cdot \sum_{i\leq n}\sum_{j\leq m} \left((V_{ij} - (WH)_{ij}) \cdot \frac{\partial}{\partial W_{ab}} (V_{ij} - (WH)_{ij}) \right) \\
&= 2 \cdot \sum_{i\leq n}\sum_{j\leq m} \left((V_{ij} - (WH)_{ij}) \cdot (-1) \cdot \frac{\partial}{\partial W_{ab}} (WH)_{ij} \right) \\
&= 2 \cdot \sum_{i=a}\sum_{j\leq m} ((WH - V)_{ij} \cdot H_{bj}) \\
&= 2 \cdot \sum_{i=a}\sum_{j\leq m} \left((WH - V)_{ij} \cdot H^T_{jb}\right) \\
&= 2 \cdot \left((WH - V)H^T\right)_{ab}
\end{aligned}$$

and derive the additive rule:

$$\begin{aligned}
W_{ab} &:= W_{ab} - \alpha_{ab} \cdot \frac{\partial}{\partial W_{ab}} e(V|WH) \\
&= W_{ab} - 2 \cdot \alpha_{ab} \cdot \left((WH - V)H^T\right)_{ab}. \qquad (1)
\end{aligned}$$

To derive a multiplicative update rule, we choose the formula for the step size to be

$$\alpha_{ab} = \frac{W_{ab}}{2 \cdot (WHH^T)_{ab}},$$

so the rule becomes as follows.

$$\begin{aligned}
W_{ab} &:= W_{ab} - 2 \cdot \frac{W_{ab}}{2 \cdot (WHH^T)_{ab}} \cdot \left((WH - V)H^T\right)_{ab} \\
&= W_{ab} - W_{ab} \cdot \frac{\left((WH - V)H^T\right)_{ab}}{(WHH^T)_{ab}} \\
&= W_{ab} \cdot \left(1 - \frac{(WHH^T)_{ab} - (VH^T)_{ab}}{(WHH^T)_{ab}}\right) \\
&= W_{ab} \cdot \frac{(VH^T)_{ab}}{(WHH^T)_{ab}} \qquad (2)
\end{aligned}$$

For H, we use the fact that

$$e(V|WH) = e(V^T|H^TW^T)$$

that switches the roles of V and V^T, W and H^T and H and W^T to get the additive update rule for H^T from (1),

$$H^T_{cb} := H^T_{cb} - 2 \cdot \beta_{cb} \cdot \left((H^TW^T - V^T)W\right)_{cb}.$$

This is equivalent to

$$\begin{aligned} H_{bc} &:= H_{bc} - 2 \cdot \beta_{bc} \cdot \left((H^TW^T - V^T)W\right)^T_{bc} \\ &= H_{bc} - 2 \cdot \beta_{bc} \cdot \left(W^T(WH - V)\right)_{bc}. \end{aligned} \tag{3}$$

Similarly, the multiplicative rule gives

$$H^T_{cb} := H^T_{cb} \cdot \frac{(V^TW)_{cb}}{(H^TW^TW)_{cb}},$$

and finally the multiplicative update rule for H,

$$H_{bc} := H_{bc} \cdot \frac{\left(W^TV\right)_{bc}}{(W^TWH)_{bc}}. \tag{4}$$

The NMF-se algorithm alternates between (2) and (4). Note that all elements in the initial matrices, including the data matrix, are assumed to be nonnegative. This implies that all three update rules multiply the previous values with nonnegative values, which means that resulting matrices W and H satisfy the constraint of nonnegativity.

Strictly speaking, the updates (2) and (4) are only defined when the denominators in them are nonzero. This has to be taken into account when implementing the algorithms. We discuss this in Section 8.1.

3.5.2 NMF update rules minimizing Kullback-Leibler divergence (NMF-kl)

In this section, our cost function is the KL divergence

$$d(V|WH) = \sum_{i \le n} \sum_{j \le m} \left(V_{ij} \cdot \ln \frac{V_{ij}}{(WH)_{ij}} - V_{ij} + (WH)_{ij} \right).$$

Derivation of the update rules for W and H follow exactly the same plot, so we only show details for W.

From Section 3.5.1, we have that the partial derivative of $(WH)_{ij}$ with respect to W_{ab} is

$$\frac{\partial}{\partial W_{ab}}(WH)_{ij} = \begin{cases} H_{bj} & \text{if } i = a, \\ 0 & \text{otherwise.} \end{cases}$$

We calculate the derivative of the cost,

$$\begin{aligned}
\frac{\partial}{\partial W_{ab}} d(V|WH) &= \frac{\partial}{\partial W_{ab}} \left(\sum_{i \le n} \sum_{j \le m} \left(V_{ij} \cdot \ln \left(\frac{V_{ij}}{(WH)_{ij}} \right) - V_{ij} + (WH)_{ij} \right) \right) \\
&= \sum_{i \le n} \sum_{j \le m} \left(V_{ij} \cdot \frac{\partial}{\partial W_{ab}} \ln \left(\frac{V_{ij}}{(WH)_{ij}} \right) + \frac{\partial}{\partial W_{ab}} (WH)_{ij} \right) \\
&= \sum_{i \le n} \sum_{j \le m} \left(V_{ij} \cdot \frac{(WH)_{ij}}{V_{ij}} \cdot V_{ij} \cdot \frac{\partial}{\partial W_{ab}} \frac{1}{(WH)_{ij}} + \frac{\partial}{\partial W_{ab}} (WH)_{ij} \right) \\
&= \sum_{i \le n} \sum_{j \le m} \left(V_{ij} \cdot (WH)_{ij} \cdot \frac{-1}{(WH)_{ij}^2} \cdot \frac{\partial}{\partial W_{ab}} (WH)_{ij} + \frac{\partial}{\partial W_{ab}} (WH)_{ij} \right) \\
&= \sum_{i \le n} \sum_{j \le m} \left(1 - \frac{V_{ij}}{(WH)_{ij}} \right) \cdot \frac{\partial}{\partial W_{ab}} (WH)_{ij} \\
&= \sum_{i = a} \sum_{j \le m} \left(1 - \frac{V_{ij}}{(WH)_{ij}} \right) \cdot H_{bj} \\
&= \sum_{j \le m} \left(1 - \frac{V_{aj}}{(WH)_{aj}} \right) \cdot H_{bj}.
\end{aligned}$$

Using the step size that Lee and Seung [35] introduced,

$$\alpha_{ab} = \frac{W_{ab}}{\sum_{j \le m} H_{bj}}.$$

gives the standard multiplicative update rule as follows.

$$
\begin{aligned}
W_{ab} &:= W_{ab} - \alpha_{ab} \cdot \frac{\partial}{\partial W_{ab}} d(V|WH) \\
&= W_{ab} - \frac{W_{ab}}{\sum_{j\le m} H_{bj}} \cdot \sum_{j\le m} \left(1 - \frac{V_{aj}}{(WH)_{aj}}\right) \cdot H_{bj} \\
&= W_{ab} \cdot \left(1 - \frac{1}{\sum_{j\le m} H_{bj}} \cdot \sum_{j\le m} \left(H_{bj} - \frac{V_{aj} \cdot H_{bj}}{(WH)_{aj}}\right)\right) \\
&= W_{ab} \cdot \left(1 - \frac{1}{\sum_{j\le m} H_{bj}} \cdot \left(\sum_{j\le m} H_{bj} - \sum_{j\le m} \frac{V_{aj} \cdot H_{bj}}{(WH)_{aj}}\right)\right) \\
&= W_{ab} \cdot \left(1 - 1 + \frac{\sum_{j\le m}(V_{aj} \cdot H_{bj})/(WH)_{aj}}{\sum_{j\le m} H_{bj}}\right) \\
&= W_{ab} \cdot \frac{\sum_{j\le m}(V_{aj}/(WH)_{aj}) \cdot H_{bj}}{\sum_{j\le m} H_{bj}} \qquad (5)
\end{aligned}
$$

The update (5) is not defined when $(WH)_{aj}$ is zero, and in Section 8.1 we discuss how this was taken into account in the implementation we used.

3.6 Sparsity constrained variants

Intuitively, sparsity means that the mass of a vector is packed to only a few elements — most elements are close to zero. In our metagenomic matrix factorization problem setting, the pathway-reaction matrix should be extremely sparse: most reactions only belong to a few pathways, and the pathways are fairly short. For example, in Section 7 we investigated MetaCyc [3] pathways implemented by human gut species and found that most pathways were less than 15 reactions long. This implies that most elements in the pathway-reaction matrix should be zero.

Sparsity constraints are often used to guide the factorization to a local minimum where the presentation of the data is parts-based. For example, in face picture recognition the base vectors define intuitive parts of the face such as eyes or nose [37]. In our task, we suppose that chemical reactions controlled by enzymes belong to small meaningful parts of organisms' metabolism, pathways. Our definition of pathways is based on MetaCyc, and these pathways are short, not holistic. They are the parts we want the representation to capture.

Some algorithms have *soft constraints*, which means that sparsity is implicitly imposed, for example, by including a sparsity term in the objective function or modifying the update rules. The reconstruction error often increases when matrices are required to be sparser, and typically the algorithms with soft constraints take parameters that control the trade-off between sparsity and reconstruction quality.

Algorithms that apply *hard constraints* take the required levels of sparsities as parameters and find a solution in the set of matrices that satisfy this constraint. The objective function is locally minimized only inside this constraint set even if significantly better reconstructions exist outside this set. Whereas soft constraints consider trade-off between reconstruction quality and sparsity, hard constraints are able to exactly determine the amount of sparsity no matter how the objective functions behave outside the constraint set.

Hard constraints fight identifiability problems by decreasing the size of the set of possible solutions. Constraints also create additional dependences between entries in the matrices, which lowers the number of free parameters.

Different definitions and measures for sparsity have been suggested in the literature, and we discuss some of these in the following sections.

3.6.1 Sparse Nonnegative Matrix Factorization (SNMF)

Liu *et al* [38] proposed Sparse Nonnegative Matrix Factorization (SNMF) and showed it yields local base images in the face recognition problem setting. They add a L^1 norm-based penalty term to the objective function, penalizing for the sum over all entries in H.

The algorithm takes one parameter α controlling trade-off between sparsity and reconstruction quality. The optimization problem SNMF solves is

$$\min_{W,H} \left\{ d(V|WH) + \alpha \cdot \sum_{k \le t} \sum_{j \le m} H_{kj} \right\},$$

given the normalizing constraint that the sum over each column in W is one. Table 3 shows the update rules of SNMF. Note that because no constraints are applied to W, the update rule for W is the NMF-kl update rule also shown in Table 2.

3.6.2 Nonnegative Matrix Factorization with sparseness constraints (NMFsc)

Hoyer [36] introduced Nonnegative Matrix Factorization with sparseness constraints, or NMFsc. It is the only variant considered in this thesis that has explicit, hard sparsity constraints. The algorithm takes two optional parameters, one deciding the amount of sparsity in W and the other in H. Constraints can specify any desired amount of sparsity

$$H_{bc} := H_{bc} \cdot \frac{\sum_{i \le n} W^T_{bi} \cdot \frac{V_{ic}}{(WH)_{ic}}}{1 + \alpha}$$

$$W_{ab} := W_{ab} \cdot \frac{\sum_{j \le m} \frac{V_{aj}}{(WH)_{aj}} \cdot H^T_{jb}}{\sum_{j \le m} H^T_{jb}}$$

Table 3: SNMF update rules

for both matrices, either one of the matrices, or neither of them. The objective function is squared error, and if no constraints are given for either matrix, the algorithm updates that matrix with standard NMF-se update rule (2) or (4).

While other variants' normalizing constraints consider L^1 norm, NMFsc sets the L^2 norm of the rows of H to one. The definition of sparsity considered in NMFsc is based on the ratio of L^1 and L^2 norms. For an m-dimensional vector $\mathbf{x}$, the definition of sparsity s is

$$s(\mathbf{x}) = \frac{\sqrt{m} - L^1(\mathbf{x})/L^2(\mathbf{x})}{\sqrt{m} - 1}. \tag{6}$$

In the case where sparsity constraints are given for W, the algorithm first updates it with the additive update rule (1). It then projects each column to the nearest vector in the Euclidean space that is nonnegative and has the L^1 norm that produces the desired value of sparsity. Similarly, when sparsity constraints are given for H, the algorithm updates H with (3), before projecting each row according to the sparsity constraint and keeping their L^2 norm set to 1.

The projection operation projects any given vector to the closest nonnegative vector that satisfies given L^1 and L^2 norm constraints. The L^1 constraint set is a hyperplane, the L^2 constraint set is a hypersphere, and the intersection of these sets is a smaller dimensional hypersphere. The projection operation first projects the input vector onto the hyperplane. It then finds the closest point of the smaller hypersphere. The L^1 and L^2 norms of this vector are as desired. In case some entries are negative, these entries are fixed to zero, and the procedure starts from the beginning, this time with the constraint that these entries are fixed to zero. Hoyer showed that, in practice, iterating this procedure quickly converges to a point where all constraints are satisfied.

```
algorithm NMFsc($V \in \mathbb{R}_{\geq 0}^{(n\times m)}$, $t \in \mathbb{N}$, $s_H \in (0,1) \cup \{\varnothing\}$)
    randomly initialize $W \in (0,1)^{(n\times t)}$ and $H \in (0,1)^{(t\times m)}$
    solve $L^1_{desired,H}$ from $s(H) = s_H$ given $L^2(H) = 1$

    for $i = 1 \to t$ do
        project (row $i$ of $H$, $L^1_{desired,H}$, 1)
    end for

    until convergence do
        $H := H - \beta \cdot W^T(WH - V)$            // this is the additive NMF-se rule (3)
        for $i = 1 \to t$ do
            project (row $i$ of $H$, $L^1_{desired,H}$, 1)
        end for
        $W := W \odot VH^T \oslash WHH^T$              // this is the multiplicative NMF-se rule (2)
    end until
end algorithm
```

algorithm ***project***($\mathbf{x} \in \mathbb{R}^k$, $L^1_{desired}$, $L^2_{desired}$): This algorithm returns the nonnegative vector with the desired L^1 and L^2 norm that is closest to $\mathbf{x}$ in the Euclidean space.

Algorithm 2: NMFsc algorithm when sparsity constraints only apply for the matrix H. Symbols $\odot$ and $\oslash$ denote element-wise product and quotient. β is a step size that the algorithm adjusts to ensure the step is small enough to decrease the cost.

The NMFsc algorithm is described in Algorithm 2.

3.6.3 Nonsmooth Nonnegative Matrix Factorization (nsNMF)

Pascual-Montano *et al* [37] proposed a method called Nonsmooth Nonnegative Matrix Factorization (nsNMF), another NMF variant with soft sparsity constraints. This method takes one parameter that controls the trade-off between imposing nonsmoothness and minimizing the reconstruction error. The goal is to make both W and H sparser than unconstrained NMF.

They modify basic NMF-kl update rules by introducing a smoothing matrix S between the two matrices W and H. If the inner dimension is t and the sparsity parameter θ, the smoothing matrix is a $(t \times t)$ symmetric matrix,

$$S_{ij} = \begin{cases} \frac{\theta}{t} + (1-\theta) & \text{if } i = j, \\ \frac{\theta}{t} & \text{otherwise.} \end{cases}$$

$$H_{bc} := H_{bc} \cdot \frac{\sum_{i \leq n} (WS)^T_{bi} \cdot \frac{V_{ic}}{(WSH)_{ic}}}{\sum_{i \leq n} (WS)^T_{bi}}$$

$$W_{ab} := W_{ab} \cdot \frac{\sum_{j \leq m} \frac{V_{aj}}{(WSH)_{aj}} \cdot (SH)^T_{jb}}{\sum_{j \leq m} (SH)^T_{jb}}$$

Table 4: nsNMF-kl update rules

$$H_{bc} := H_{bc} \cdot \frac{\left((WS)^T V\right)_{bc}}{((WS)^T WSH)_{bc}}$$

$$W_{ab} := W_{ab} \cdot \frac{\left(V(SH)^T\right)_{ab}}{(W(SH)(SH)^T)_{ab}}$$

Table 5: nsNMF-se update rules

The modified update rules are similar to basic NMF-kl update rules, except that in the update rule for W, the matrix H is substituted by the product SH and in the update rule for H, the matrix W is substituted by the product WS. This is equivalent to minimizing the extended KL divergence between the data and the product WSH, and we denote this nsNMF algorithm by nsNMF-kl.

These rules make the fixed matrix appear smoother to the matrix that is being updated, and the update rule then compensates this smoothness by making the updated matrix sparser.

We applied their strategy also in the case of Euclidean squared error, and denote the resulting algorithm by nsNMF-se. Update rules for nsNMF-se result from same substitutions to basic NMF-se update rules. Tables 4 and 5 show update rules for both algorithms.

$$\begin{bmatrix}0.3 & 0.2 & 0.2 & 0 & 0.2 & 0.1\\ 0.3 & 0.1 & 0 & 0.2 & 0.4 & 0\\ 0.4 & 0.2 & 0.2 & 0 & 0.1 & 0.1\\ 0.2 & 0.2 & 0.2 & 0 & 0.1 & 0.3\end{bmatrix} \approx \begin{bmatrix}0.2 & 0.7 & 0.1\\ 0.9 & 0.1 & 0\\ 0 & 1 & 0\\ 0 & 0.6 & 0.4\end{bmatrix} \underbrace{\begin{bmatrix}0.3 & 0.1 & 0 & 0.2 & 0.4 & 0\\ 0.4 & 0.2 & 0.2 & 0 & 0.1 & 0.1\\ 0 & 0.1 & 0.2 & 0 & 0.2 & 0.5\end{bmatrix}}_{\substack{\approx \\ \overbrace{\begin{bmatrix}1 & 0\\ 1.3 & 0.7\\ 0 & 1\end{bmatrix}\begin{bmatrix}0.3 & 0.1 & 0 & 0.2 & 0.3 & 0\\ 0 & 0.1 & 0.2 & 0 & 0.05 & 0.4\end{bmatrix}}}}$$

left-hand approach: $V \approx W(HP)$

$$\begin{bmatrix}0.3 & 0.2 & 0.2 & 0 & 0.2 & 0.1\\ 0.3 & 0.1 & 0 & 0.2 & 0.4 & 0\\ 0.4 & 0.2 & 0.2 & 0 & 0.1 & 0.1\\ 0.2 & 0.2 & 0.2 & 0 & 0.1 & 0.3\end{bmatrix} \approx \underbrace{\begin{bmatrix}1.1 & 0.6\\ 1 & 0.1\\ 1.3 & 0.7\\ 0.8 & 0.8\end{bmatrix}}_{\substack{\approx \\ \overbrace{\begin{bmatrix}0.2 & 0.7 & 0.1\\ 0.9 & 0.1 & 0\\ 0 & 1 & 0\\ 0 & 0.6 & 0.4\end{bmatrix}\begin{bmatrix}1 & 0\\ 1.3 & 0.7\\ 0 & 1\end{bmatrix}}}} \begin{bmatrix}0.3 & 0.1 & 0 & 0.2 & 0.3 & 0\\ 0 & 0.1 & 0.2 & 0 & 0.05 & 0.4\end{bmatrix}$$

right-hand approach: $V \approx (WH)P$

Figure 2: Three-matrix factorizations by factorizing the input twice into two matrices

4 Factorization into Three Matrices in Two Steps

In our problem setting, our goal is to factorize our input sample-reaction matrix into three matrices: sample-OTU, OTU-pathway and pathway-reaction matrices.

The most straightforward implementation of a Nonnegative Matrix Factorization variant that factorizes the input into three matrices instead of two is to run NMF twice. In this approach, the data matrix is first factorized into two matrices, and factorizing one of the resulting matrices again into two gives a factorization into three matrices: $V \approx AP$ and $A \approx WH$ give $V \approx WHP$. Note that also factorizations $V \approx WB$ and $B \approx HP$ give $V \approx WHP$. We call this latter strategy the *left-hand approach* because the leftmost matrix W is learned first. Similarly, we call the factorization in the first example the

$$W_{ab} := W_{ab} \cdot \sqrt{\frac{(VP^TH^T)_{ab}}{(WW^TVP^TH^T)_{ab}}}$$

$$H_{bc} := H_{bc} \cdot \frac{(W^TVP^T)_{bc}}{(W^TWHPP^T)_{bc}}$$

$$P_{cd} := P_{cd} \cdot \frac{(H^TW^TV)_{cd}}{(H^TW^TVP^TP)_{cd}}$$

Table 6: Update rules of NMTF

right-hand approach. Both strategies are shown in Figure 2.

Some NMF variants are better suited for extracting particular kinds of features from the data. When factorizing the data in two steps, different variants can be used to perform each of the factorizations.

The downside is, because both factorizations are approximate, the overall cost between V and WHP is not necessarily minimized. In our problem setting, the left-hand approach means we first extract OTUs from the data and use the resulting OTU-reaction matrix to learn pathways. Now, when we take the left-hand approach, it is not possible to take into account the pathways when predicting OTUs, because we learn and fix the matrix that defines OTUs first. Similarly, the right-hand approach cannot exploit any OTU information when predicting pathways.

This motivates an extension that learns all three matrices simultaneously, as illustrated in Figure 3. Such algorithms are discussed in Sections 5 and 6.

5 Nonnegative Matrix Tri-Factorization (NMTF)

In 2006, Ding *et al* [40] proposed a nonnegative matrix factorization method called NMTF that factorizes the data matrix into three matrices simultaneously. Their algorithm is designed for document clustering and has hard orthogonality constraints on the first and third matrices of the factorization.

As with standard NMF, $V \in \mathbb{R}^{(n \times m)}$ denotes a given data matrix, t_1 and t_2 desired inner dimensions. NMTF is defined as the constrained optimization problem of finding a minimum of $e(V|WHP)$, so that $W \in \mathbb{R}^{(n \times t_1)}$, $H \in \mathbb{R}^{(t_1 \times t_2)}$, $P \in \mathbb{R}^{(t_2 \times m)}$ and each

$$\boxed{\begin{bmatrix} 0.3 & 0.2 & 0.2 & 0 & 0.2 & 0.1 \\ 0.3 & 0.1 & 0 & 0.2 & 0.4 & 0 \\ 0.4 & 0.2 & 0.2 & 0 & 0.1 & 0.1 \\ 0.2 & 0.2 & 0.2 & 0 & 0.1 & 0.3 \end{bmatrix} \approx \begin{bmatrix} 0.2 & 0.7 & 0.1 \\ 0.9 & 0.1 & 0 \\ 0 & 1 & 0 \\ 0 & 0.6 & 0.4 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1.3 & 0.7 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.3 & 0.1 & 0 & 0.2 & 0.3 & 0 \\ 0 & 0.1 & 0.2 & 0 & 0.05 & 0.4 \end{bmatrix}}$$

Figure 3: Three-matrix factorization: $V \approx WHP$

element of W, H and P is nonnegative. Additionally, W and P are constrained to be orthogonal, *i.e.* the products W^TW and PP^T must equal the identity matrix.

The NMTF algorithm iteratively updates one matrix at a time keeping the other two matrices fixed. They note that existing update rules for the corresponding orhogonality constrained 2-matrix algorithm can be applied for updating the first and the third matrix by replacing the fixed matrix in the rule with the matrix product of the two fixed matrices; for example, in the standard update rule for W, H is replaced with HP. This rule simply updates the weights in W to optimally represent the data using the given base vectors, regardless of whether the base vectors are given in one matrix H, or as a product of two matrices, HP. This idea resembles the idea used in nsNMF, discussed in Section 3.6.3.

Updating the matrix H requires some additional adjustments: the algorithm has to take into account both of the fixed matrices at the same time. This requires exploiting information coming from the left side of the matrix as well as from the right side simultaneously. Ding *et al* give an update rule for H and prove the correctness and convergence of this rule. The cost function they consider is squared error, and the updates rules are shown in Table 6. We show derivation of this update rule using the gradient descent method in Section 6.

6 Native Three-matrix Factorization (NMF3)

As before, let $V \in \mathbb{R}^{(n \times m)}$ be a given data matrix and $c : \mathbb{R}^{(n \times m)} \times \mathbb{R}^{(n \times m)} \to \mathbb{R}$ a differentiable cost function. Let t_1 and t_2 be desired inner dimensions. We define three-matrix factorization as the constrained optimization problem of finding a local minimum of $c(V|WHP)$, so that $W \in \mathbb{R}^{(n \times t_1)}$, $H \in \mathbb{R}^{(t_1 \times t_2)}$, $P \in \mathbb{R}^{(t_2 \times m)}$ and each element of W, H and P is nonnegative.

We consider the squared error and KL divergence cost functions. We denote these variants by NMF3-se and NMF3-kl.

Similarly to the NMTF algorithm, we keep two matrices fixed while updating one. In the following, we assume all three matrices W, H and P are initialized with nonnegative

$V \approx WHP$	W	H	P
NMF3-se	-	rows	rows
NMF3-kl	-	rows	rows
SNMF3	columns	columns	-

$V \approx WB$	W	B	
NMF-se	columns	-	
NMF-kl	columns	-	

$A \approx WH$	W	H	
NMF-se	-	rows	
NMF-kl	-	rows	

$V \approx AP$ or $B \approx HP$		A or H	P
SNMF		columns	-
NMFsc		-	rows, L^2
NMF-se		-	rows
NMF-kl		-	rows
nsNMF-kl		-	rows
nsNMF-se		-	rows

Table 7: Normalization constraints of NMF and NMF3 variants that we used in our experiments. All algorithms apart from NMFsc have L^1-based constraints *i.e.* the sum over each column or row is one. The sample-pathway matrix produced in the first step of the two-step algorithms is denoted by A and the OTU-reaction matrix by B.

real values, the data matrix is a nonnegative $(n \times m)$ matrix, and the required inner dimensions are t_1 and t_2.

Ding *et al* [40] claim that the three factor NMF is "only interesting when it can not be transformed into two-factor NMF", referring to a situation when WH in the three-factor setting can be directly mapped to W in the two-factor setting. For example, in the case the inner dimensions are the same, $t_1 = t_2$, the unconstrained three-matrix NMF problem can actually be solved with a corresponding two-matrix variant by adding the $t_1 \times t_1$ identity matrix between the resulting two matrices. To avoid this, Ding *et al* apply hard orthogonality constraints on two of the matrices. However, we implemented unconstrained three-factor algorithms NMF3-se and NMF3-kl and tested them on our data to succesfully recover features of our interest. Results are shown in Section 8.

In theory, three-matrix variants are more susceptible to identifiability difficulties than two-matrix variants. We apply similar normalization constraints to the ones used in

$$W_{ab} := W_{ab} \cdot \frac{(VP^TH^T)_{ab}}{(WHPP^TH^T)_{ab}}$$

$$H_{bc} := H_{bc} \cdot \frac{(W^TVP^T)_{bc}}{(W^TWHPP^T)_{bc}}$$

$$P_{cd} := P_{cd} \cdot \frac{(H^TW^TV)_{cd}}{(H^TW^TWHP)_{cd}}$$

Table 8: Update rules for the unconstrained 3-matrix extension of NMF minimizing squared error, NMF3-se. The update rule for H is the same as in NMTF, shown in Table 6.

corresponding two-matrix variants. We scale the rows of P and either the rows of H or columns of W by exploiting the fact that

$$WHP = W(D_1D_1^{-1})H(D_2D_2^{-1})P$$

holds for any diagonal matrices D_1 and D_2, similarly as discussed in Section 3.4 in the case of two-matrix variants.

All normalization constraints we used are shown in Table 7. Note that the constraints can be changed by taking the transpose of the data matrix and the resulting matrices. For example, the row-wise normalization of H and P in NMF3-se changes to column-wise normalization of W and H. Note that NMF3-se and NMF3-kl can easily be modified to normalize columns of W instead of rows of H, which results in a variant where no constraints are applied to H.

6.1 Update rules minimizing squared error (NMF3-se)

We derive multiplicative update rules for our 3-matrix NMF extension. They are shown in Table 8. The goal is to approximately factorize given data matrix V into three matrices, $V \approx WHP$. In this subsection, we consider the squared error cost function $e(V|WHP) = \sum_{i \le n} \sum_{j \le m} (V_{ij} - (WHP)_{ij})^2$. Derivation of the update rules for all three matrices follow the same pattern, and we first show details for the second matrix H.

First, we note that by the definition of matrix product the product of three matrices is defined by

$$(WHP)_{ij} = \sum_{l \leq t_2} (WH)_{il} \cdot P_{lj} = \sum_{l \leq t_2} \left(\sum_{k \leq t_1} W_{ik} \cdot H_{kl} \right) \cdot P_{lj} = \sum_{l \leq t_2} \sum_{k \leq t_1} W_{ik} \cdot H_{kl} \cdot P_{lj}.$$

In the following, we need partial derivatives of elements of the product WHP, so we calculate

$$\begin{aligned} \frac{\partial}{\partial H_{bc}} (WHP)_{ij} &= \sum_{l \leq t_2} \sum_{k \leq t_1} \frac{\partial}{\partial H_{bc}} (W_{ik} \cdot H_{kl} \cdot P_{lj}) \\ &= \sum_{l \leq t_2} \Big(\sum_{k < b} \frac{\partial}{\partial H_{bc}} (W_{ik} \cdot H_{kl} \cdot P_{lj}) + \sum_{k = b} \frac{\partial}{\partial H_{bc}} (W_{ik} \cdot H_{kl} \cdot P_{lj}) \\ &\quad + \sum_{b < k \leq t_1} \frac{\partial}{\partial H_{bc}} (W_{ik} \cdot H_{kl} \cdot P_{lj}) \Big). \end{aligned}$$

Now, because the derivative of $(W_{ik} \cdot H_{kl} \cdot P_{lj})$ with respect to H_{bc} is zero when $k \neq b$, we can simplify the above formula to

$$\begin{aligned} \frac{\partial}{\partial H_{bc}} (WHP)_{ij} &= \sum_{l \leq t_2} \sum_{k = b} \frac{\partial}{\partial H_{bc}} (W_{ik} \cdot H_{kl} \cdot P_{lj}) \\ &= \sum_{l < c} \frac{\partial}{\partial H_{bc}} (W_{ib} \cdot H_{bl} \cdot P_{lj}) + \sum_{l = c} \frac{\partial}{\partial H_{bc}} (W_{ib} \cdot H_{bl} \cdot P_{lj}) \\ &\quad + \sum_{c < l \leq t_2} \frac{\partial}{\partial H_{bc}} (W_{ib} \cdot H_{bl} \cdot P_{lj}), \end{aligned}$$

and similarly, with respect to H_{bc}, the derivative of $(W_{ib} \cdot H_{bl} \cdot P_{lj})$ is zero when $l \neq c$, which gives

$$\frac{\partial}{\partial H_{bc}} (WHP)_{ij} = \frac{\partial}{\partial H_{bc}} (W_{ib} \cdot H_{bc} \cdot P_{cj}) = W_{ib} \cdot P_{cj}.$$

Next, we calculate the derivative of the cost function with respect to H_{bc},

$$\begin{aligned}
\frac{\partial}{\partial H_{bc}} e(V|WHP) &= \sum_{i\leq n}\sum_{j\leq m} \frac{\partial}{\partial H_{bc}} \left(V_{ij} - (WHP)_{ij}\right)^2 \\
&= 2\cdot \sum_{i\leq n}\sum_{j\leq m} \left((V_{ij} - (WHP)_{ij}) \cdot \frac{\partial}{\partial H_{bc}} (V_{ij} - (WHP)_{ij}) \right) \\
&= 2\cdot \sum_{i\leq n}\sum_{j\leq m} \left((V_{ij} - (WHP)_{ij}) \cdot (-1) \cdot \frac{\partial}{\partial H_{bc}} (WHP)_{ij} \right) \\
&= 2\cdot \sum_{i\leq n}\sum_{j\leq m} \left((WHP - V)_{ij} \cdot W_{ib} \cdot P_{cj} \right) \\
&= 2\cdot \sum_{i\leq n}\sum_{j\leq m} \left(W^T_{bi} \cdot (WHP - V)_{ij} \cdot P^T_{jc} \right) \\
&= 2\cdot (W^T(WHP - V)P^T)_{bc}.
\end{aligned}$$

Finally, setting the step size β_{bc} in the gradient descent rule to equal

$$\beta_{bc} = \frac{H_{bc}}{2\cdot (W^TWHPP^T)_{bc}},$$

gives us the multiplicative update rule

$$\begin{aligned}
H_{bc} &:= H_{bc} - \beta_{bc} \cdot \frac{\partial}{\partial H_{bc}} e(V|WHP) \\
&= H_{bc} - \frac{H_{bc}}{2\cdot (W^TWHPP^T)_{bc}} \cdot 2 \cdot (W^T(WHP - V)P^T)_{bc} \\
&= H_{bc} - \frac{H_{bc} \cdot ((W^TWHPP^T)_{bc} - (W^TVP^T)_{bc})}{(W^TWHPP^T)_{bc}} \\
&= H_{bc} \cdot \frac{(W^TVP^T)_{bc}}{(W^TWHPP^T)_{bc}}. \qquad (7)
\end{aligned}$$

To derive the update rule for W, we calculate the derivative of the cost with respect to W_{ab}. Following similar steps as before, we start by calculating the partial derivative of

$(WHP)_{ij}$.

$$\frac{\partial}{\partial W_{ab}}(WHP)_{ij} = \frac{\partial}{\partial W_{ab}} W_{ib} \cdot (HP)_{bj} = \begin{cases} (HP)_{bj} & \text{if i=a,} \\ 0 & \text{otherwise} \end{cases}$$

Then, we calculate the derivative of the cost.

$$\begin{aligned} \frac{\partial}{\partial W_{ab}} e(V|WHP) &= 2 \cdot \sum_{i=a} \sum_{j \le m} ((WHP - V)_{ij} \cdot (HP)_{bj}) \\ &= 2 \cdot ((WHP - V)(HP)^T)_{ab} \end{aligned}$$

In order to derive a multiplicative update rule from the additive rule, we choose the step size

$$\alpha_{ab} = \frac{W_{ab}}{2 \cdot (WHPP^T H^T)_{ab}},$$

so the multiplicative rule is as follows.

$$\begin{aligned} W_{ab} &:= W_{ab} - \alpha_{ab} \cdot \frac{\partial}{\partial W_{ab}} e(V|WHP) \\ &= W_{ab} \cdot \frac{(VP^T H^T)_{ab}}{(WHPP^T H^T)_{ab}}. \end{aligned} \quad (8)$$

The above rule is the same update rule that is obtained by replacing the matrix H with the matrix product HP in the standard NMF multiplicative update rule. To find the update rule for P, we note that taking the transpose of WHP switches the roles of W and P^T, and the update rule for W gives a similar update for P when the transpose is translated back. Because we have already derived the update rule for the first matrix (8), we can use the symmetric nature of the factorization, and the fact that

$$e(V|WHP) = e(V^T|(WHP)^T) = e(V^T|P^T H^T W^T),$$

to reason that the multiplicative update rule for P^T is

$$P^T_{dc} := P^T_{dc} \cdot \frac{(V^T WH)_{dc}}{(P^T H^T W^T WH)_{dc}}.$$

This means the update rule for P is

$$P_{cd} := P_{cd} \cdot \frac{(V^T WH)^T_{cd}}{(P^T H^T W^T WH)^T_{cd}} = P_{cd} \cdot \frac{(H^T W^T V)_{cd}}{(H^T W^T WHP)_{cd}}. \tag{9}$$

Again, the above rule is the same that is attained by replacing W with the product WH in the standard NMF update rule.

Our three-matrix NMF algorithm NMF3-se alternates between (7), (8) and (9). Note that all elements in initial matrices, including the data matrix, are assumed to be nonnegative. This implies that all three update rules multiply the previous values with nonnegative values, which means that resulting matrices W, H and P satisfy the constraint of nonnegativity.

As mentioned in Section 3.5.1, also the updates (7-9) are only defined when the denominators in them are nonzero. Details of the implementation are discussed in Section 8.1.

6.2 Update rules minimizing KL divergence (NMF3-kl)

In this section, we derive multiplicative update rules for NMF3-kl, our NMF extension that approximately factorizes given data matrix V into three matrices, $V \approx WHP$, minimizing Kullback-Leibler divergence from the data matrix V to the reconstructed matrix WHP. The update rules are shown in Table 9 We now assume all three matrices W, H and P are initialized with *positive* values, the data matrix is $(n \times m)$ matrix, and the required inner dimensions are t_1 and t_2. We first derive the update rule for the second matrix H. From the previous section, we have the partial derivatives

$$\frac{\partial}{\partial H_{bc}}(WHP)_{ij} = W_{ib} \cdot P_{cj}.$$

$$W_{ab} := W_{ab} \cdot \frac{\displaystyle\sum_{j \le m} \frac{V_{aj}}{(WHP)_{aj}} \cdot (HP)^T_{jb}}{\displaystyle\sum_{j \le m} (HP)^T_{jb}}$$

$$H_{bc} := H_{bc} \cdot \frac{\displaystyle\sum_{i \le n} \sum_{j \le m} W^T_{bi} \cdot \frac{V_{ij}}{(WHP)_{ij}} \cdot P^T_{jc}}{\displaystyle\sum_{i \le n} W^T_{bi} \cdot \sum_{j \le m} P^T_{jc}}$$

$$P_{cd} := P_{cd} \cdot \frac{\displaystyle\sum_{i \le n} (WH)^T_{ci} \cdot \frac{V_{id}}{(WHP)_{id}}}{\displaystyle\sum_{i \le n} (WH)^T_{ci}}$$

Table 9: Update rules for the 3-matrix extension of NMF minimizing KL divergence, NMF3-kl

We use this to calculate the derivative of the cost function with respect to H_{bc},

$$\begin{aligned}
\frac{\partial}{\partial H_{bc}} d(V|WHP) &= \frac{\partial}{\partial H_{bc}} \sum_{i\leq n} \sum_{j\leq m} \left(V_{ij} \cdot \ln\left(\frac{V_{ij}}{(WHP)_{ij}}\right) - V_{ij} + (WHP)_{ij} \right) \\
&= \sum_{i\leq n} \sum_{j\leq m} V_{ij} \cdot \frac{\partial}{\partial H_{bc}} \ln\left(\frac{V_{ij}}{(WHP)_{ij}}\right) + \frac{\partial}{\partial H_{bc}} (WHP)_{ij} \\
&= \sum_{i\leq n} \sum_{j\leq m} V_{ij} \cdot \frac{(WHP)_{ij}}{V_{ij}} \cdot V_{ij} \cdot \frac{\partial}{\partial H_{bc}} \frac{1}{(WHP)_{ij}} + \frac{\partial}{\partial H_{bc}} (WHP)_{ij} \\
&= \sum_{i\leq n} \sum_{j\leq m} -\frac{V_{ij}}{(WHP)_{ij}} \cdot \frac{\partial}{\partial H_{bc}} (WHP)_{ij} + \frac{\partial}{\partial H_{bc}} (WHP)_{ij} \\
&= \sum_{i\leq n} \sum_{j\leq m} \left(1 - \frac{V_{ij}}{(WHP)_{ij}}\right) \cdot \frac{\partial}{\partial H_{bc}} (WHP)_{ij} \\
&= \sum_{i\leq n} \sum_{j\leq m} \left(1 - \frac{V_{ij}}{(WHP)_{ij}}\right) \cdot W_{ib} \cdot P_{cj}.
\end{aligned}$$

Now, setting the step size β_{bc} in the additive rule to equal

$$\beta_{bc} = \frac{H_{bc}}{\sum\limits_{i\leq n} \sum\limits_{j\leq m} W_{ib} \cdot P_{cj}},$$

gives us the multiplicative update rule

$$\begin{aligned}
H_{bc} &:= H_{bc} - \beta_{bc} \cdot \frac{\partial}{\partial H_{bc}} d(V|WHP) \\
&= H_{bc} - \frac{H_{bc}}{\sum\limits_{i\leq n} \sum\limits_{j\leq m} W_{ib} \cdot P_{cj}} \cdot \sum_{i\leq n} \sum_{j\leq m} \left(1 - \frac{V_{ij}}{(WHP)_{ij}}\right) \cdot W_{ib} \cdot P_{cj} \\
&= H_{bc} \cdot \left(1 - 1 + \frac{\sum_{i\leq n} \sum_{j\leq m} W_{ib} \cdot \frac{V_{ij}}{(WHP)_{ij}} \cdot P_{cj}}{\sum\limits_{i\leq n} \sum\limits_{j\leq m} W_{ib} \cdot P_{cj}} \right) \\
&= H_{bc} \cdot \frac{\sum_{i\leq n} \sum_{j\leq m} W^T_{bi} \cdot (V_{ij}/(WHP)_{ij}) \cdot P^T_{jc}}{\sum\limits_{i\leq n} \sum\limits_{j\leq m} W^T_{bi} \cdot P^T_{jc}}. \qquad (10)
\end{aligned}$$

To derive the rule for updating W, we calculate the derivative of the cost with respect to W_{ab}. From section 6.1, we have that the partial derivative of $(WHP)_{ij}$ is

$$\frac{\partial}{\partial W_{ab}}(WHP)_{ij} = \begin{cases} (HP)_{bj} & \text{if } i = a, \\ 0 & \text{otherwise,} \end{cases}$$

and we calculate the derivative of the cost,

$$\begin{aligned}
\frac{\partial}{\partial W_{ab}} d(V|WHP) &= \frac{\partial}{\partial W_{ab}} \left(\sum_{i\leq n} \sum_{j \leq m} \left(V_{ij} \cdot \ln \left(\frac{V_{ij}}{(WHP)_{ij}} \right) - V_{ij} + (WHP)_{ij} \right) \right) \\
&= \sum_{i\leq n} \sum_{j \leq m} \left(1 - \frac{V_{ij}}{(WHP)_{ij}} \right) \cdot \frac{\partial}{\partial W_{ab}} (WHP)_{ij} \\
&= \sum_{i=a} \sum_{j \leq m} \left(1 - \frac{V_{ij}}{(WHP)_{ij}} \right) \cdot (HP)_{bj} \\
&= \sum_{j \leq m} \left(1 - \frac{V_{aj}}{(WHP)_{aj}} \right) \cdot (HP)_{bj}.
\end{aligned}$$

We use the step size introduced in [35], but replace H with HP,

$$\alpha_{ab} = \frac{W_{ab}}{\sum\limits_{j \leq m} (HP)_{bj}}.$$

This again gives a multiplicative update rule:

$$\begin{aligned}
W_{ab} &:= W_{ab} - \alpha_{ab} \cdot \frac{\partial}{\partial W_{ab}} d(V|WHP) \\
&= W_{ab} - \frac{W_{ab}}{\sum_{j\leq m} (HP)_{bj}} \cdot \sum_{j \leq m} \left(1 - \frac{V_{aj}}{(WHP)_{aj}} \right) \cdot (HP)_{bj} \\
&= W_{ab} \cdot \left(1 - 1 + \frac{\sum_{j \leq m} (V_{aj} \cdot (HP)_{bj})/(WHP)_{aj}}{\sum_{j\leq m} (HP)_{bj}} \right) \\
&= W_{ab} \cdot \frac{\sum_{j\leq m} (V_{aj}/(WHP)_{aj}) \cdot (HP)^T_{jb}}{\sum_{j \leq m} (HP)^T_{jb}}. \qquad (11)
\end{aligned}$$

This rule is the same update rule that is obtained by replacing the matrix H with the matrix product HP in the standard NMF multiplicative update rule suggested in [37]. Derivation of the update rule for P follows exactly the same pattern that was shown for W and H - calculating partial derivatives of the cost function and then setting the step size for P_{cd} to equal $P_{cd}/\sum_{i\leq n}(WH)_{ic}$. Without showing details, we mention that the end result is the same that results from substituting W by WH in the standard rule for NMF-kl,

$$P_{cd} := P_{cd} \cdot \frac{\sum_{i\leq n}(WH)^T_{ci} \cdot V_{id}/(WHP)_{id}}{\sum_{i\leq n}(WH)^T_{ci}}. \tag{12}$$

In the beginning of this section we assumed that elements of the estimation WHP are initially positive. The extended KL divergence is not defined if it happens that for some i and j, $(WHP)_{ij} = 0$ but $V_{ij} > 0$. In this case, we interpret that the estimate is infinitely far from representing the data. As all values are nonnegative in every iteration of the algorithm, the elements of WHP are linear combinations where both bases and coefficients are nonnegative. This means that zeros in the product can only be produced by zeros in the matrices W, H and P. Because our update rules are multiplicative, updating has no effect on zeros, which means that the "inifinitely bad" estimate cannot be updated so that the cost would decrease to a finite value.

Like mentioned in Section 3.5.2, also the updates (10-12) are only defined when the denominators in them are nonzero. Details of the implementation are again discussed in Section 8.1.

$$W_{ab} := W_{ab} \cdot \frac{\displaystyle\sum_{j \le m} \frac{V_{aj}}{(WHP)_{aj}} \cdot (HP)^T_{jb}}{\displaystyle\sum_{j \le m} (HP)_{bj}}$$

$$H_{bc} := H_{bc} \cdot \frac{\displaystyle\sum_{i \le n} \sum_{j \le m} W^T_{bi} \frac{V_{ij}}{(WHP)_{ij}} \cdot P^T_{jc}}{\displaystyle\sum_{i \le n} W_{ib} \cdot \sum_{j \le m} P_{cj}}$$

$$P_{cd} := P_{cd} \cdot \frac{\displaystyle\sum_{i \le n} (WH)^T_{ci} \cdot \frac{V_{id}}{(WHP)_{id}}}{1 + \alpha}$$

Table 10: Update rules for the 3-matrix extension of SNMF

6.3 Update rules for 3-matrix Sparse NMF (SNMF3)

In SNMF, the first matrix W is updated by standard NMF-kl update rule (5), and H is updated with a rule that incorporates the sparsity constraint specified by parameter α, as shown in Table 3 on page 20. The three-matrix variant of this algorithm should impose sparsity on the third matrix P, and we can use the NMF3-kl update rules derived in Section 6.2 to update the other matrices.

In the factorization $V \approx WHP = (WH)P$ we again think of the matrix product of the first two matrices WH as one matrix, and the update rule for P results from SNMF rule for H by susbtituting the matrix H with P and then W with the matrix product WH. The SNMF3 update rules are shown in Table 10.

SNMF has the normalization constraint that the sum over each column of W is one, and in SNMF3, we correspondingly normalize columns of both W and H.

7 Data Set Construction

Validation methods based on knowing the underlying structure of the data cannot be applied when we use real-life data that are not fully understood. We studied different algorithms' behaviour on artificial data to be able to apply validation methods. These data should both resemble real data and contain latent variables of interest. Most importantly, we have to ensure we know the ground truth in order to evaluate the algorithms' results.

We used the Human Microbiome Project's [7] knowledge of gut species together with MetaCyc's [3] pathway and enzyme information to make our data realistic. This serves our purpose of developing methods that are compatible with the characteristics of real metagenomic data. Control of the generating process gives us a means of systematic evaluation.

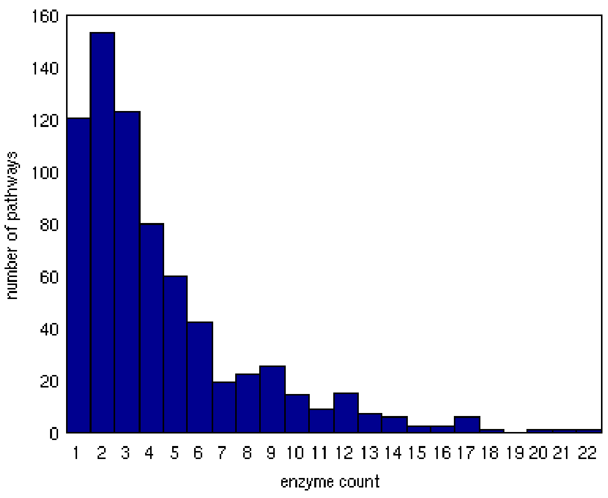


Figure 4: Pathway sizes in selected 100 OTUs

We started with all microbial species from Human Microbiome Project that were assigned to the body site 'gastrointestinal tract' and discarded all but one strain of each species. We then used MetaCyc to collect lists of pathways implemented by each species, discarding all species that were not found in MetaCyc or were not assigned any pathways. We grouped species that were assigned exactly the same sets of pathways into *Operational Taxonomical Units* (OTUs). This level of taxonomy corresponds to the lowest level where each OTU implements a unique set of pathways. The result is a list of 218 OTUs from which we took a random subset of $t_1 = 100$ OTUs.

We again used MetaCyc to collect lists of *Enzyme Commission* (EC) numbers of enzymes that are present in each pathway. Because each pathway should be defined by a unique set of enzymes, we only kept one pathway for each distinct enzyme set. The total number of pathways that the 100 OTUs implement was 710. To avoid uninterestingly short pathways, or including exceptionally long super pathways, we discarded pathways that were less than three or more than 14 enzymes long. The upper threshold of 14 enzymes is based on Figure 4 that shows longer pathways were rare in the original set. The result is a set of $t_2 = 422$ pathways and a total of 1234 enzymes.

7.1 Artificial data

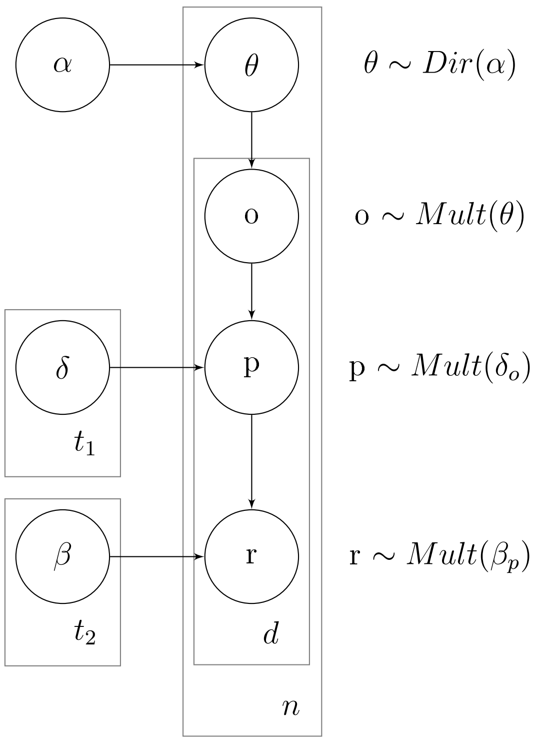
Our artificial data is based on the OTU-pathway and pathway-reaction information described above. We represent the lists of pathways that each of the 100 OTUs implement with a binary 100×422 matrix whose element (k, l) indicates whether pathway l is assigned to OTU k. By dividing each element (k, l) of this matrix by the sum over row k, the rows are normalized and become distributions. Correspondingly, all 422 pathways are defined as sets of reactions in a binary 422×1234 matrix. Similarly, this matrix is normalized row-wise, which results in a row-matrix of distributions where each row defines one pathway.

We constructed $n = 1500$ samples. For each sample $i \leq n$, an OTU distribution θ_i is drawn from a Dirichlet distribution. These distributions are collected in a 1500×100 matrix so that each row i gives the distribution of OTUs over sample i. The set of OTUs is the same set of 100 OTUs in all samples in a given data set, but their proportions vary. We use the parameter vector $\alpha = (\alpha_1, \alpha_2, ..., \alpha_{t_1}) = (0.1, 0.1, ..., 0.1)$, *i.e.* $\alpha_k = 0.1$ for all k. This gives very peaked distributions with a few dominant OTUs per sample. For example, the average number of OTUs in a sample that have probabilities lower than 0.001 is 66, and for probabilities greater than 0.02 the number is 12.45, which means that on average around 66 out of the 100 OTUs have very low abundances while only around a dozen OTUs are dominant.

Now, we take the matrix product of the above mentioned sample-OTU, OTU-pathway and pathway-reaction matrices. Each entry (i, j) is a linear combination of pathways with coefficients in the OTU-pathway matrix, weighed by the sample-OTU distribution of sample i. Thus, the entry (i, j) is a real number that represents a relationship between the sample i and reaction j. We say that this relationship depicts the amount of evidence of the reaction j that the sample i includes, and the rows of this matrix product are our artificial metagenomic samples. Hence, we call this 1500×1234 matrix our *artificial data matrix*. We normalize each row of the matrix, because it is a standard procedure with real metagenomic samples [21, 20].

Each subset $\mathcal{S} \subset \{1, 2, ..., 1500\}$ of rows of the artificial data matrix defines a smaller data set $\mathcal{D}$ with fewer samples. If we instead take the corresponding subset of the sample-OTU distributions, and multiply the resulting sample-OTU submatrix with the same OTU-pathway and pathway-reaction matrices as before, the result is exactly the same matrix $\mathcal{D}$. We randomly took such subsets of sizes 500 and 100 to produce two additional data matrices, 500×1234 and 100×1234, so we can compare the methods' performance in cases where different amount of samples is available.

Note that in the constructing process of the data matrix, the distributions of pathways over OTUs as well as reactions over pathways result from normalizing binary matrices, in contrast to the distributions of OTUs over samples which were drawn from a Dirichlet



α	Dirichlet parameter vector
θ	defines the sample-OTU distribution
δ	defines OTU-pathway distributions
t_1	total number of available OTUs
β	defines pathway-reaction distributions
t_2	total number of available pathways
o	observed OTU
p	observed pathway
r	observed reaction
d	sampling depth
n	number of samples

Figure 5: Data generative model

distribution. Hence, the nature of the data in the first matrix is different from the second and third matrices. This was taken into account when we chose suitable validation methods, discussed in Section 8.2.

7.2 Data generative model

We defined a generative model that is able to capture the parameters we want to extract from metagenomic data and used the model to simulate synthetic data. The model is shown in Figure 5, and the sampling process based on the model is described in Algorithm 3.

As with artificial data, $n = 1500$ samples in total were generated from the model and the resulting (1500×1234) sample-reaction matrix are the input data for our experiments on simulated data — our *simulated data matrix*.

In the sampling process, we used the same sample-OTU, OTU-pathway and pathway-reaction distribution matrices as with artificial data described in Subsection 7.1. We set the number of observations, or *sampling depth*, d to 1000, which means the total count of reaction observations is 1000 per sample. This means we cannot observe more than 1000

```
**algorithm** ***GenerateDataset***($n$, $\delta$, $\beta$)
    initialize sample-reaction matrix $V$ with zeros
    **for** $i = 1 \to n$ **do**
        sample-otu distribution $\theta \sim$ Dir(0.1)      // Pick the distribution according to a Dirichlet
        **repeat** 1000 times
            OTU o $\sim$ Mult($\theta$)      // Pick according to the sample-otu distribution
            pathway p $\sim$ Mult($\delta_o$)      // Pick according to the otu-specific pathway distribution
            reaction r $\sim$ Mult($\beta_p$)      // Pick according to the pathway-specific reaction distribution
            $V_{ir} \leftarrow V_{ir} + 1$      // Observe the selected reaction
        **end repeat**
    **end for**
    **return** $V$
**end algorithm**
```

Algorithm 3: Generative process based on the model in Figure 5

different reactions per sample, despite that all 1234 reactions have a probability greater than zero to be picked for each row. On average, less than 500 different reactions were observed, because many reactions were observed more than once. The simulated data matrix is not normalized, but the sum over each row is 1000.

Limited sampling depth makes the simulated data set very different from the artificial set. For example, because the sample-OTU distributions are peaked, for each sample the dominant OTUs are observed often and possibly some less abundant OTUs are not observed at all. In artificial data, each sample has at least some evidence of every single reaction: every element of the artificial data matrix is positive, whereas 62% of the elements of the simulated data matrix are zeros. In this sense, the sampling depth of the artificial data set is infinite, and this allows us to study the algorithms' performance in a hypothetical situation where sampling depths of metagenomic data are arbitrarily large.

8 Results

We test the NMF algorithms' performance in two different settings. The first task is to factorize a known matrix product and study different algorithms' behaviour in a setting where an exact factorization exists. This completely artificial, noise-free data can be accurately factorized, unlike actual data. We select a few methods for the second task based on their performance on artificial data.

Our second task is to infer the species and pathways from the data that was generated using the sampling process described in Section 7. In this setup, there is some noise due to sampling, and we assume the factorization to be approximate. We use the initial distributions to validate the results from this task.

To study the impact of the data set size, we test all algorithms on sets of 1500, 500 and 100 samples, *i.e.*, rows of the data matrix. Currently, no metagenomic data sets of as many as 1500 samples are available. This largest set represents the optimal case where both inner dimensions of the factorization are smaller than the dimensions of the data. With 500 samples, the number of pathways is only 15.6% smaller than the smaller dimension of the data.

Theoretically, the factorization is not expected to behave well when the inner dimension of the factorization is the same or even smaller than dimensions of the data. We use the set of 100 samples to study how the algorithms behave in this very difficult task.

8.1 Implementation

We first studied how standard, unconstrained NMF algorithms behave in our task. We have used Matlab code provided in Hoyer's software package *nmfpack* [36] with slight modifications[1]. We consider NMF variants using two popular choices of cost function, Euclidean squared error and KL divergence, and denote these variants by NMF-se and NMF-kl. We further modified the software into NMF3-se and NMF3-kl, described in Sections 6.1 and 6.2. These algorithms factorize the input matrix into three matrices.

For sparsity constrained variants, we modified implementations of Sparse Nonnegative Matrix Factorization (SNMF), [38], and Nonnegative Matrix Factorization with sparseness constraints (NMFsc) found in *nmfpack*. We further modified this source code into SNMF3 described in Section 6.3. We implemented nsNMF-se [37] and nsNMF-kl, described in Section 3.6.3, by adding the smoothing matrix to the update rules of NMF-se and NMF-kl.

As mentioned in the previous chapters, many updates are only defined when denominators in them are nonzero. Our implementation is based on *nmfpack*, so we mostly use

[1]We only use iteration count as a stopping criterion instead of manually monitoring the convergence. We modify the term $V_{ij}/(WH)_{ij}$ to avoid undefined expressions in the update rules of some algorithms.

their strategy and add a small constant 10^{-9} to the problematic denominators. We found that SNMF and SNMF3 were sometimes unable to process input matrices that had too many zeros. To avoid this problem, we substitute the value of V_{ij}/WHP_{ij} with 1 when $V_{ij} = WHP_{ij} = 0$, because the value of this quotient is 1 also whenever $V_{ij} = WHP_{ij} \neq 0$. This strategy differs from the above mentioned one where only a small constant is added to all denominators, because when the numerator is zero, the quotient evaluates to 0. This makes values in the updated matrices decrease very steeply when the input V contains many zeros.

For the factorization schemes where the data matrix is factorized by running two-matrix NMF variants twice, we first run all the algorithms for the first step. In the left-hand approach, we then have all the factorizations into sample-OTU and OTU-reaction matrices. We evaluate the sample-OTU matrices and select the factorization instance where this matrix is qualitatively best *i.e.* the instance that learned most *agreed OTUs* as defined in Section 8.2. In cases where there were more than one such factorization, we selected the one with most learned OTUs according to KL divergence. The corresponding OTU-reaction matrix then serves as an input for all algorithms' second step, where pathways are extracted from this OTU-reaction matrix.

Similarly, in the right-hand approach we evaluate the resulting pathway-reaction matrices and select the sample-pathway matrix that corresponds to the qualitatively best one *i.e.* the one with most *agreed learned pathways*, also defined in Section 8.2. If there is more than one such factorization, we choose the one with more learned pathways when measured using only the absolute error.

Because the input matrix is the same for all algorithms, we are able to accurately compare the algorithms' performance in the following step. Although selecting qualitatively best factorizations might not be possible when real-life data is used, this best serves our purpose of comparing the methods.

8.1.1 Initialization

As discussed in Section 3, the algorithms only find local optima of the objective. Starting from some initial point, they iteratively update the matrices. The choice of this initial point has an effect on which local minimum is found. For simplicity, we initialized all matrices with pseudorandom values drawn uniformly from the open interval $(0, 1)$. Randomness helps to avoid the possible problem of starting the factorization from the most unfavorable points of the space.

Because the matrices to which the update rules are applied are already scaled in every other iteration of the algorithms, we also normalize rows or columns of the initial matrices in accordance with the normalization constraints of the algorithm in question. Table 7 shows the normalization constraints of each of the algorithms.

We ran every setup of the algorithms ten times, on different initial matrices each time. In Sections 8.3 and 8.4 we report the results that show that initialization causes some variation on most algorithms' performance.

8.1.2 Stopping criteria

We decided to set a number of iterations for each algorithm as the only stopping criterion. We do not monitor convergence, but the numbers that we used are based on our observations on how many iterations the algorithms usually took to converge on our data. These observations are based on visual interpretations of the convergence curve, *i.e.*, how the values of the objective function change through iterations of the algorithm.

We found that all of the two-matrix variants converged in at most 500 iterations. To make sure they would converge properly, we set the number of iterations to 600. On our data, the three-matrix variants converged slower, and the number of iterations was set to 5000 to ensure convergence.

8.1.3 Parameters

Parameter selection for all algorithms that use soft constraints usually has to be done empirically. Some algorithms' parameters can only take values in a given set, such as the open interval $(0, 1)$. Other algorithms make parameter selection even more inconvenient as the parameters can take, for example, any positive real value.

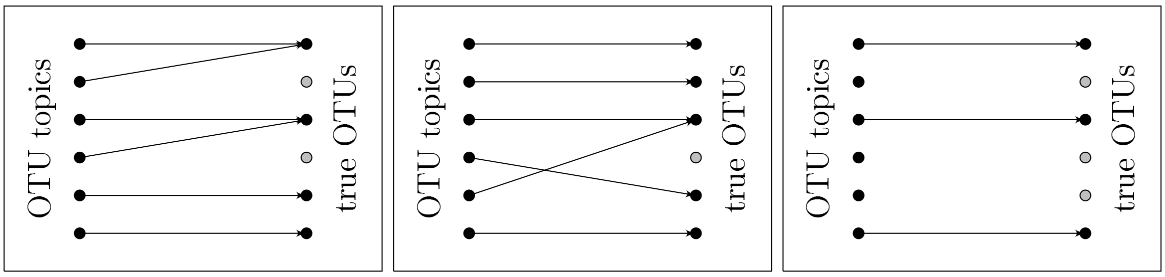
We chose the sparsity parameters 10, 50, 100 and 200 for SNMF and SNMF3. According to the original paper [38], the parameter 100 should make the resulting matrix H 100 times sparser than unconstrained NMF. For both nonsmooth variants nsNMF-se and nsNMF-kl, we chose the sparsity parameters 0.2, 0.4, 0.6 and 0.8.

The hard constraint of NMFsc lets us explicitly select how sparse the matrix P becomes. The sparsity of the underlying matrix was 0.962, and based on this we used sparsity parameters 0.9, 0.95 and 0.962.

8.2 Evaluation metrics

In this section, we describe the metrics and measures we considered when evaluating the methods' performances, and explain how we visualize the results. Our goal is to find methods that best suit metagenomic type data.

Absolute error between n-dimensional vectors $\mathbf{x}$ and $\mathbf{y}$ is defined as the sum over the



a) absolute error *b) KL divergence* *c) these metrics agree*

Figure 6: Learned OTUs. Arrows show what true OTUs are represented by the learned OTU topics. In *a* and *b*, each arrow points to the true OTU from which the absolute error or KL divergence to a given OTU topic is minimal. Graph *c* shows true OTUs that absolute error and KL divergence match to the same OTU topic. In this example, the number of learned OTUs is 4 if measured by absolute error (a) and 5 if KL divergence is used (b) and the number of agreed learned OTUs is 3 (c).

absolute values of the differences of their elements,

$$\sum_{i \leq n} |\mathbf{x}_i - \mathbf{y}_i|.$$

Number of learned pathways and OTUs. We call rows in the initial pathway-reaction distribution matrix *true pathways*, and by *pathway topics* we mean normalized *rows* in the third matrix that the algorithms produce. For each pathway topic p', we interpret that p' represents the true pathway p for which a chosen cost function from p to p' is smallest. For a factorization $V \approx WHP$, the number of learned pathways is the number of different true pathways that were represented by pathway topics in P.

We apply the same idea to evaluate how many *true OTUs* are represented by *OTU topics* in the *columns* of the first matrix of the factorization. In this case, we first normalize the columns in the initial sample-OTU distribution matrix to obtain true OTUs. Results discussed in more detail in Subsection 8.3.1 show that neither KL divergence nor absolute error alone was able to distinguish randomly generated sample-OTU matrices from matrices that we learn with NMF techniques. We observed similar behaviour for the number of learned pathways, discussed in Section 8.3.2.

However, when we only take into account OTU topics that represent the same true OTU in terms of both KL divergence and absolute error, *i.e.*, both measures assign the OTU topic to the same true OTU, we get another number: the number of *agreed* learned OTUs. Figure 6 shows an example. Mathematically, if we denote the set of OTU topics by $\mathcal{S}$ and the set of true OTUs by $\mathcal{O}$ the number is the size of the set

$$\{o | \exists s \in \mathcal{S} \text{ so that } o = \arg\min_{o_k \in \mathcal{O}} d(o_k|s) = \arg\min_{o_k \in \mathcal{O}} b(o_k|s)\}.$$

This measure both requires the true OTU to be point-wise the nearest true OTU and that as distributions the divergence from this true OTU is smallest among all true OTUs. It is more strict than absolute error or KL divergence alone. We use the same strategy to calculate the number of agreed pathways.

This evaluation metric is not designed to measure the accuracy with which each pathway or OTU was learned, but reflects qualitative features of the learned topics in one number. If the algorithm separates all true pathways or OTUs from each other, the number of learned pathways or OTUs equals the number of the true ones. Similarly, if the algorithm has extracted some other features from the data than the ones in which we are interested, we expect this number to be low. To examine this more closely, we have conducted tests with random matrices described below.

The order of the topics in NMF methods does not carry any meaning: any permutation of them gives exactly the same reconstruction of the data. The order is mostly defined by the initialization of the matrices, which in our case is random. Importantly, this metric is able to evaluate matrices where the learned topics are in an arbitrary order.

Finding the closest matches also allows us to align learned topics to true features and correspondingly arrange the result matrices. This further allows us to apply more general evaluation methods such as receiver operating characteristics, described below, for pathway data. Importantly, the quality of the second matrix can only be evaluated after aligning OTU and pathway topics. Each row of this matrix corresponds to one OTU topic, and each column to a pathway topic, and entries describe their relationships predicted by our learning method. We must evaluate the sample-OTU and pathway-reaction matrices and align the learned topics to true OTUs and pathways in order to examine how well their relationships were predicted.

Random matrices. It is not clear how the chosen evaluation metrics, number of learned pathways and OTUs, behave with random guesses. We do not know what the expected numbers are if we analyze randomly generated matrices. To investigate this, we draw ten matrices of 1500 species distributions of 100 species each from a Dirichlet distribution using parameter $\alpha = 0.1$. This gives us ten sample-species distribution matrices generated exactly the same way than the original ground truth matrix W. We include these matrices in the analysis as if they were learned matrices.

Note that the generating process of the random matrices includes the knowledge of the Dirichlet parameter that was used to generate underlying species features of the data, and that we do not use this information when learning these features from the sample-reaction matrix with NMF techniques.

The generative process for random pathway matrices is as follows. In Section 7, we mention the distribution of pathway lengths in the initial pathway-reaction matrix. We draw lengths of at least 3 and at most 14 for 422 random pathways from this distribution, and uniformly sample a corresponding number of reactions for each pathway. To be able to calculate KL divergences, we add a small constant 10^{-9} to all elements of this binary matrix. For comparison to learned pathway matrices, we generate ten matrices using this procedure and evaluate them similarly.

Receiver Operating Characteristics (ROC), and the Area Under the Curve (AUC). The nature of our pathways is binary: a given enzyme either belongs to a given MetaCyc pathway or does not. We evaluate the quality of the learned pathways by calculating the ROC curves and the area under them — standard methods for evaluating binary classifiers. The advantage of this is that the AUC sums up the quality of the learned matrix in one number; the downside is that binarizing discards some information.

We have a binary pathway-reaction matrix P_{true} and its estimate P_{estim} whose entries are real values between zero and one. We fix a threshold α and binarize the estimate P_{estim} by substituting all values smaller than α by zero, and larger values by 1. From this binary estimate matrix, we can count what values were predicted correctly.

True Positive Rate (TPR) is the number of ones that were predicted correctly by P_{estim} divided by the total number of ones in P_{true}. False Positive Rate (FPR) is the number of zeros that were erroneously predicted to be ones divided by the total number of zeros in P_{true}. ROC curve shows the TPR as a function of FPR when the threshold α goes through values from one to zero.

With a large threshold α, most values in the binarized estimate are zeros, and both TPR and FPR are near zero. As the threshold is decreased, the number of ones in the estimate increases. If the estimate is perfect *i.e.* there exists a threshold that binarizes the estimate to exactly P_{true}, FPR stays at zero as TPR goes to one. When α reaches zero, all values in the estimate are ones, and both TPR and FPR are 1. The more area there is between the ROC curve and the x-axis, the better the estimate is considered to be.

Box plot. We use the Matlab box plot with default values to show the number of learned OTUs and pathways. Mathworks website [42] explains the box plot: "On each box, the central mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually."

Chessboard plot. To visually investigate the quality of our result matrices, we show chessboard plots of cost matrices between learned topics and true features. In each such chessboard plot, the rows correspond to learned topics and the columns corerspond to true OTUs or true pathways. The color of the element (i, j) shows the KL divergence from true OTU j to OTU topic i, or the squared error between true pathway j and pathway topic i.

We align the learned topics with true OTUs or pathways by permuting both the rows and columns of the cost matrix. The first row and first column are selected so that the smallest dissimilarity in the matrix goes in the upper left corner. This means we align the most accurately learned feature first. After this first permutation, we fix the first row and column, find the smallest element of the remaining matrix, and permute the remaining matrix so that this smallest value goes to element (2,2). We continue similarly until we have ordered the whole matrix.

This greedy alignment is well suited for investigating how many features were extracted and how accurately they were learned. In cases where two or more learned topics represent the same underlying feature, the most accurate one is aligned to that feature and the other topics are aligned to some other features, possibly ones that are not well represented by any learned topic. Features that are combined in one topic, or split into different topics are not taken into account as only the pairwise similarities between one topic and one feature are considered at a time.

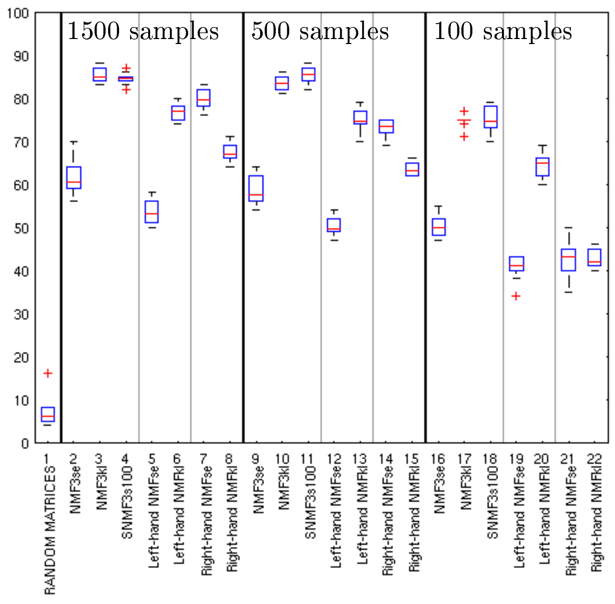


Figure 7: Number of agreed OTUs each of the algorithms learned from 1500, 500 and 100 artificial samples

8.3 Experiments on completely artificial data

In this section, we study the performance of our methods on artificial data described in Section 7.1. During our experiments, no algorithm factorized the data exactly into the initial matrices. This highlights the fact that the algorithms typically converge to local minima of the objective.

We first ran a series of experiments on unconstrained algorithms. We found that KL divergence-based variants were able to extract OTU features from the data. However, the pathway-reaction matrix that we used to build artificial data was extremely sparse. None of the matrices we produced with NMF-kl, NMF-se or the corresponding three-matrix variants NMF3-kl and NMF3-se were as sparse, so we used the sparsity constrained variants introduced in Sections 3.6.1 - 3.6.3 and 6.3 to impose sparsity on the pathway-reaction matrix. The sparsity of the matrix P produced with different algorithms is shown in Figure 12 and is discussed in Section 8.3.2.

8.3.1 OTUs from artificial data

The highest agreed number of OTUs that the algorithms recovered was very high, 90 out of the total 100, and the mean of the numbers of OTUs both NMF3-kl and SNMF3 recovered from 1500 and 500 samples was around 85. On 100 samples, the numbers are lower and the results vary more. This suggests that the factorization is less stable when the inner dimension, *i.e.* the number of OTU topics, is the same as the number of samples. Still, we were able to recover many OTUs also in the case where only this small number

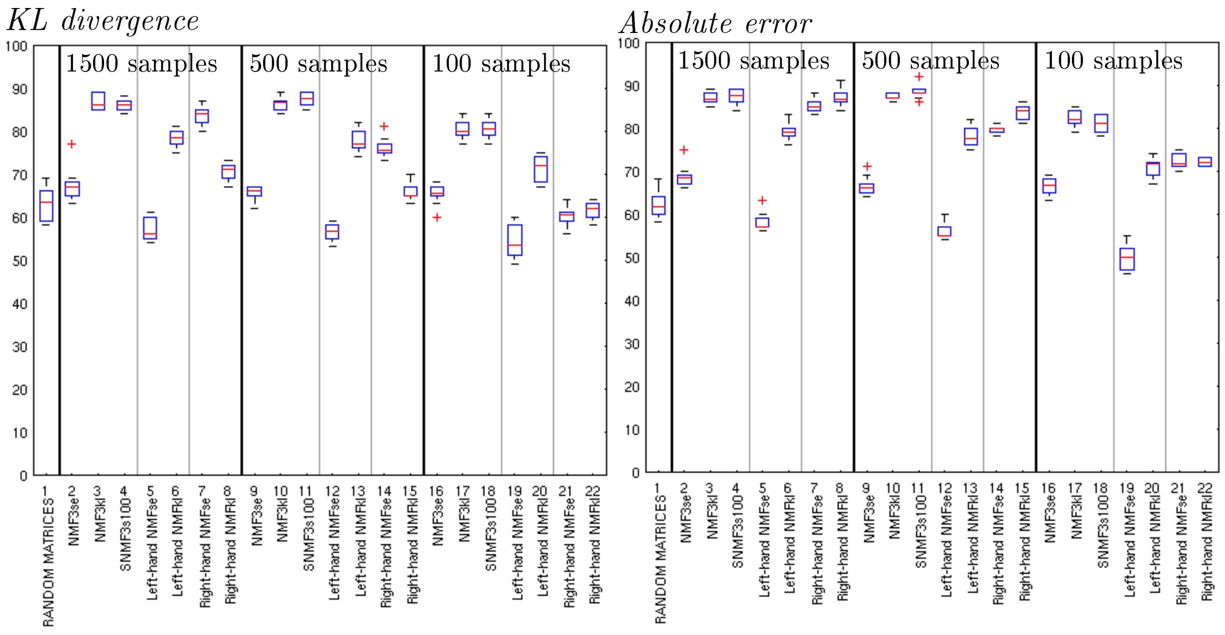


Figure 8: Number of OTUs each of the algorithms learned from 1500, 500 and 100 artificial samples when only one of the measures is considered

of samples is available.

Figures 7 and 8 show the number of learned OTUs from three artificial data sets of different sizes, using three different measures. Especially the agreed number of learned OTUs shows clear difference between randomly generated matrices and all the NMF techniques shown in Figure 7. Apart from the right-hand factorization, the results from all algorithms are along similar lines on 500 samples as they are on 1500 samples. It is interesting that the relations between different algorithms remained roughly the same even when a very low number of samples, 100, was used.

The KL divergence-based variants NMF-kl, NMF3-kl and SNMF3 were able to recover more OTUs than NMF-se and NMF3-se in all other settings except the right-hand approach on 1500 and 500 samples. Regardless of what measure is used – KL divergence, absolute error or the agreed number – the number of OTUs KL divergence-based algorithms extracted is higher than the corresponding number for randomly generated matrices. We expect the difference between the variants that minimize different objective functions due to the nature of the underlying features. KL divergence is better suited for comparing distributions, and the underlying OTUs are defined by Dirichlet distributions. The L^1 norm-based sparsity constraint that SNMF3 has on the matrix P does not affect the number of learned OTUs compared to unconstrained NMF-kl.

Native three-matrix factorization algorithms performed better than their two-matrix counterparts. This shows that the native method is able to benefit from extracting both features at the same time.

We chose one individual factorization per data set size for further analysis. On 1500 samples, the highest number of agreed learned OTUs was 89, by NMF3-kl. However, the number of learned pathways was low as shown in Section 8.3.2. As a compromise, we chose the factorization where the sum of agreed learned OTUs and agreed learned pathways was highest. On 1500, 500, as well as 100 samples this was achieved with left-hand factorization approach by NMF-kl for the first step. The input data for the next step of the left-hand factorization are the OTU-reaction matrices resulting from these same individual factorizations. For these chosen factorizations, the number of agreed learned OTUs was 80 on 1500 samples, 79 on 500 samples and 69 on 100 samples.

For the chosen three individual factorizations, Figure 9 shows KL divergence from each true OTU to each OTU topic. Rows and columns are permuted as described in Section 8.2. If the factorization was perfect, all elements on the diagonal would be black, indicating zeros, and all other elements would be orange or yellow.

Figure 9 (a) shows that when all 1500 samples were used, we learned around 60 OTUs very well. The values on the diagonal are visibly higher in the lower right corner between OTUs 80 and 100, which shows the connection between the number of learned OTUs and the chessboard plot as the number of agreed learned OTUs was 80. Figure 9 (d) shows a sub-matrix of the KL divergence cost matrix with only the rows and columns where KL divergence and absolute error align the row with the same column. The plot shows that by considering only these agreed alignments we can discard less accurately learned OTUs.

From all three data sets, the most accurately learned true OTU is the same. This OTU implements 154 pathways. Only 13 OTUs out of the total 100 OTUs implement more pathways. There are 6 pathways that are unique to this OTU, out of 53 pathways that are unique to one OTU.

We aligned the learned OTUs of all three factorizations using three different measures: in addition to KL divergence and Euclidean squared error, we have aligned cost matrices that consider the absolute error. There were three true OTUs that were among the 20 last columns in all cases. We interpret that none of the three factorization instances in question learned these OTUs accurately. All pathways that any of these three OTUs implement are also implemented by some other OTU. They implement 95, 127 and 135 pathways, while for comparison only 21 out of all 100 OTUs implement less than 95 pathways.

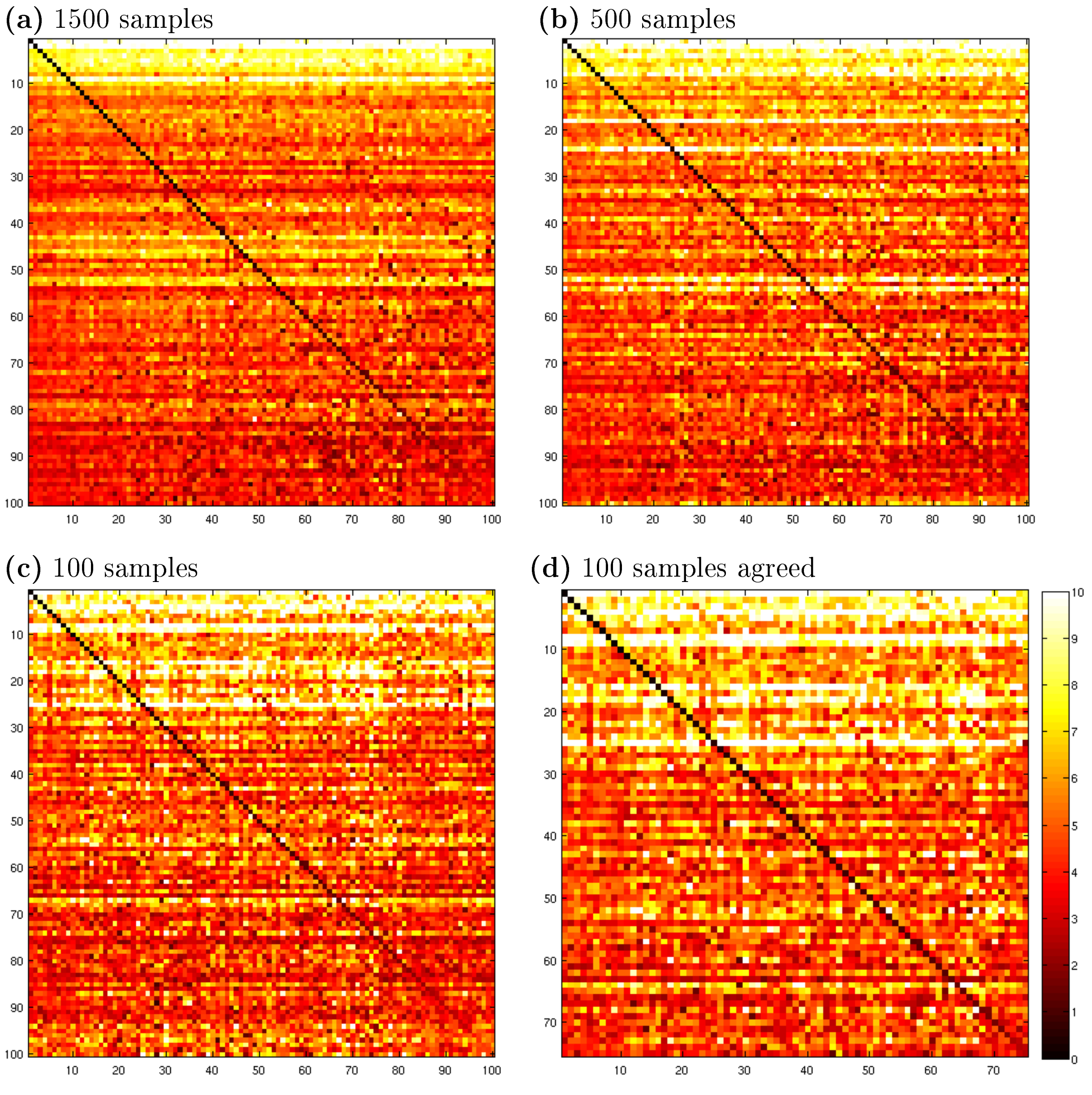
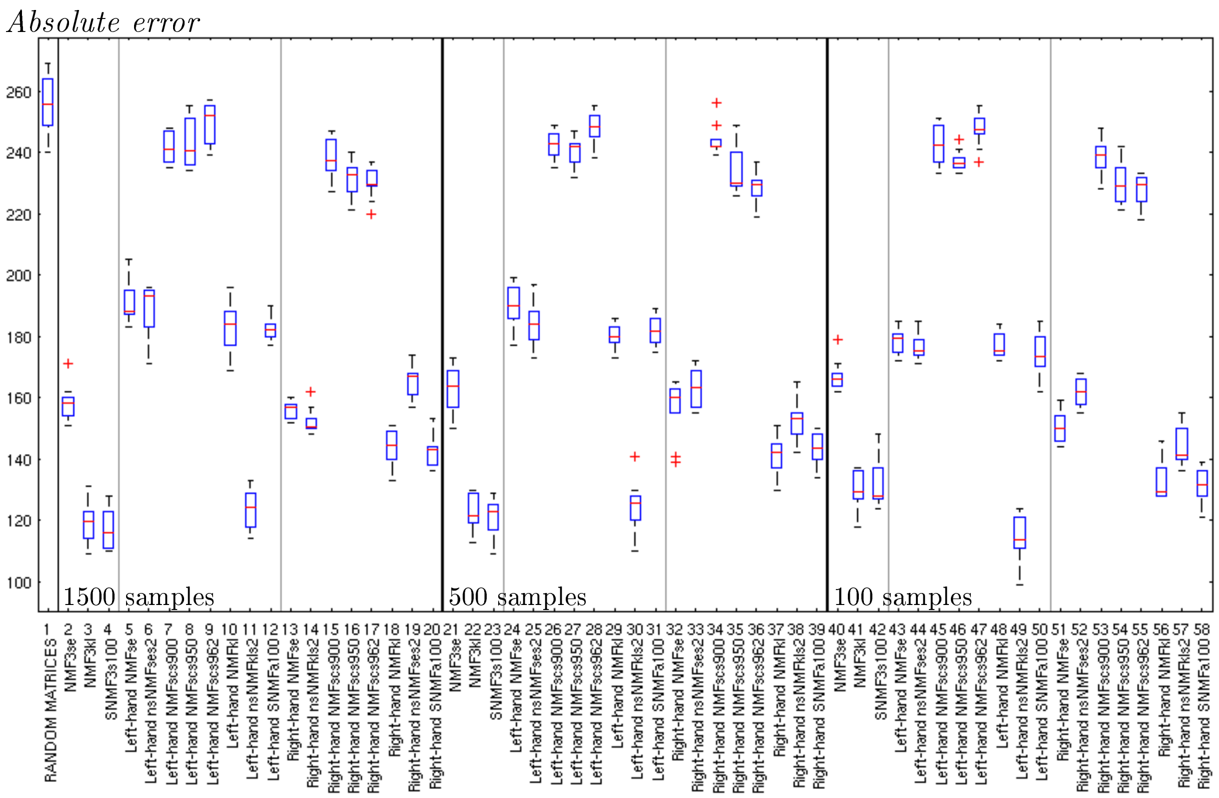


Figure 9: Chessboard plot from qualitatively good sample-OTU matrices after aligning the learned topics with true OTUs. The colour of element (i, j) shows the *KL divergence* from true OTU j to OTU topic i. Higher divergences than 10 are displayed using white colour.



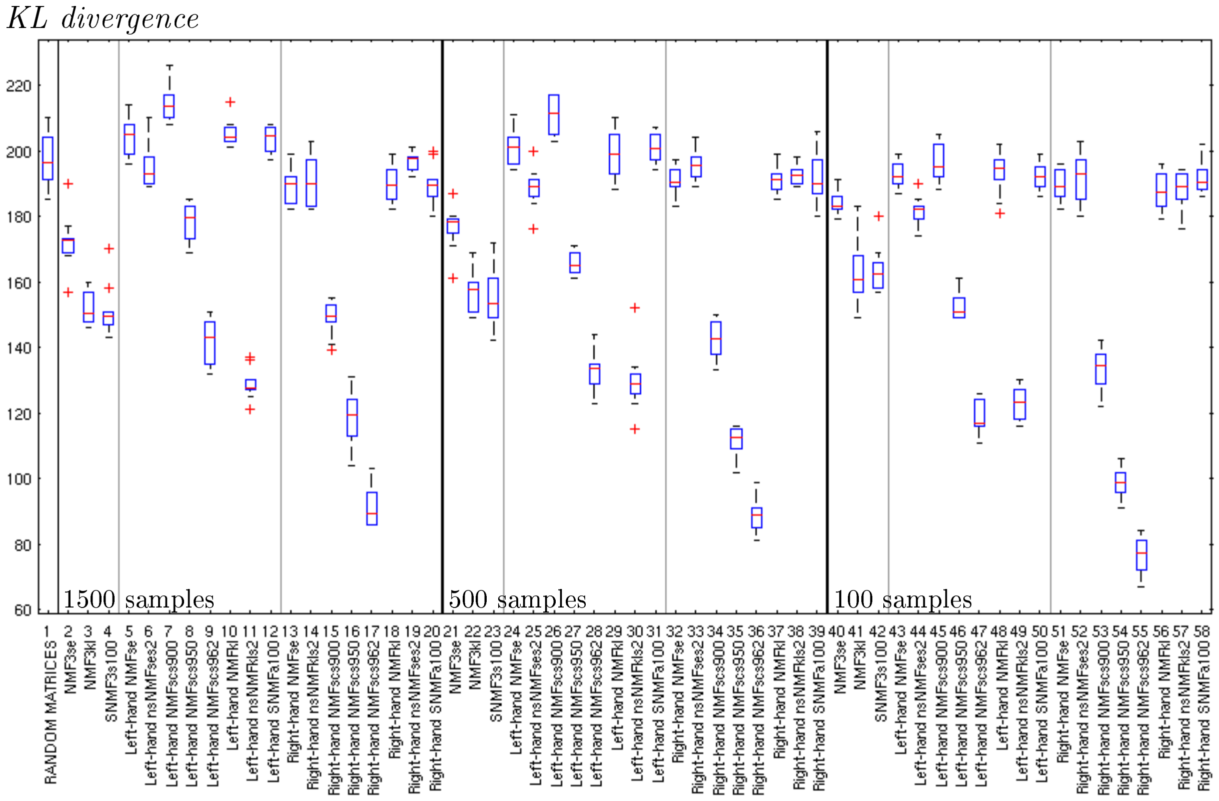


Figure 10: Number of pathways each of the algorithms learned from 1500, 500 and 100 samples measured using only one of the metrics at a time

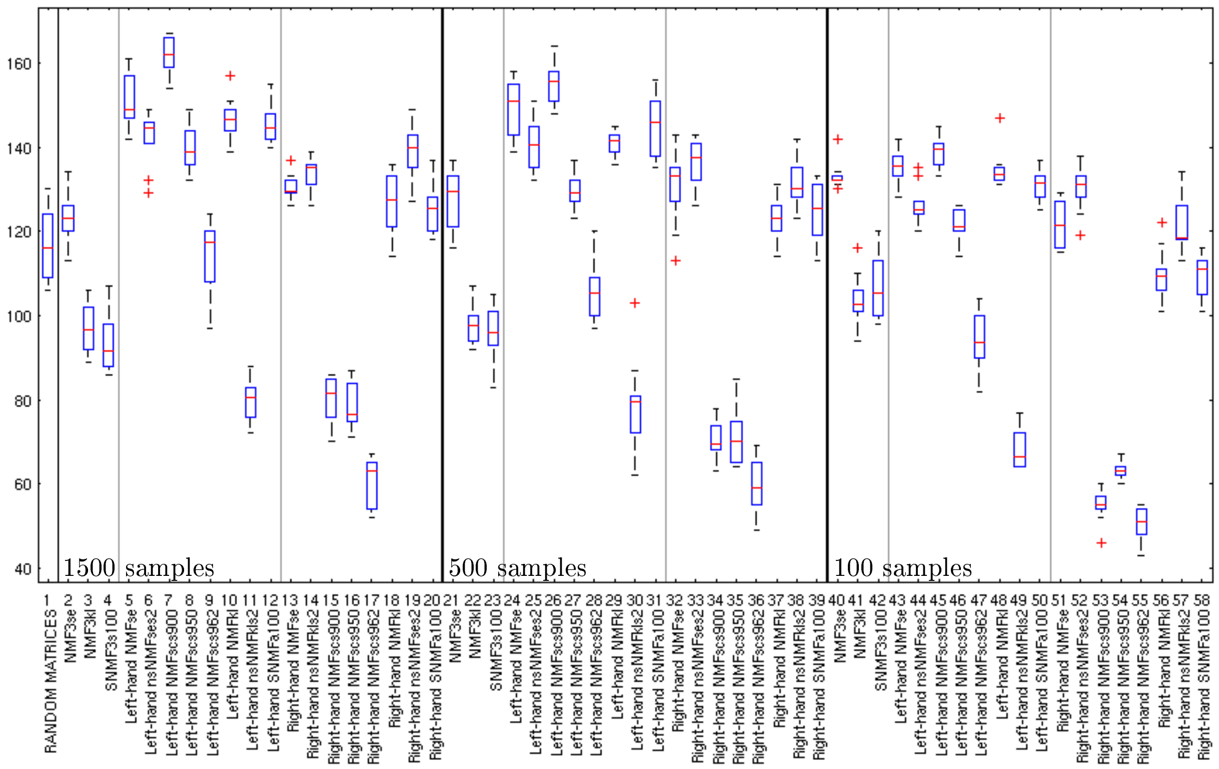


Figure 11: Number of agreed pathways each of the algorithms learned from 1500, 500 and 100 samples. These are the numbers of pathway topics that KL divergence and absolute error align to the same real pathways.

8.3.2 Pathways from artificial data

Figure 10 shows that both absolute error and KL divergence assign high numbers of learned pathways to randomly generated control matrices. However, Figure 11 reveals that when both measures are taken into account at the same time, most of our factorization techniques extracted more pathways than what was found in the control matrices.

Figures 10-11 show that the performance of each algorithm in recovering pathways is very similar regardless of which data set size was used. Whereas KL divergence-based algorithms were better suited for extracting OTU information, squared error-based algorithms NMF-se and NMFsc with parameter 0.9 in general recover more pathways than KL divergence-based algorithms in the left-hand setup, and nsNMF-se in the right-hand setup. We expect this due to the nature of the underlying data discussed in Section 7.

Most algorithms perform better in the left-hand setup in which pathways are learned from the OTU-reaction matrix produced in the first factorization. This is very interesting because the input OTU-reaction matrix is very small: the number of OTUs is only 100, less than one quarter of the number of pathways. Still, many algorithms were able to recover more than 140 pathways.

Surprisingly, the performance of all three-matrix variants is better when fewer samples is used but not as good as many of the two-matrix algorithms'. The performance of both SNMF and SNMF3 is not very different from the corresponding unconstrained variant

NMF-kl or NMF3-kl. We interpret that the L^1 norm-based sparsity constraint of SNMF3 is not useful for extracting pathway features.

Instead, we have further investigated the affect of L^2 norm-based sparsity defined by Equation (6), and Figure 12 shows how sparse the result matrices of different variants are when ran on 1500 samples. Corresponding box plots on 500 and 100 samples (figures not included) show very similar behaviour than Figure 12.

Overall, left-hand approach results in sparser matrices than right-hand approach. This offers another explanation why the left-hand approach yielded better results. The right-hand factorizations factorize the 1500×1234, 500×1234 or 100×1234 sample-reaction data matrix whereas the input matrix in the left-hand factorization is the 100×1234 OTU-reaction matrix. The size of the input matrix is not solely responsible for the sparsity of the output matrices which can be seen from the results on 100 samples where the input is the same size on both approaches.

Due to the different definition of sparsity, there is no visible difference between the L^2 norm-based sparsity of SNMF or SNMF3 and the corresponding unconstrained NMF-kl or NMF3-kl. Based on the number of learned pathways measured using any of the three measures, the L^1 norm-based sparsity constraints of SNMF and SNMF3 do not significantly affect the method's qualitative performance compared to NMF-kl or NMF3-kl, respectively.

Figure 11 shows that NMFsc with sparsity parameter 0.9 recovered more pathways than any other variant. Because of hard sparsity constraints on P, NMFsc also produces the sparsest pathway matrices. Although with parameter value 0.962 the sparsity of the matrix is almost the same as the sparsity of the ground truth matrix, the number of agreed learned pathways is smaller than with the parameter 0.9. This might be because extreme parameters strongly restrict the constraint set, and the fit of the reconstruction to the data is compromised.

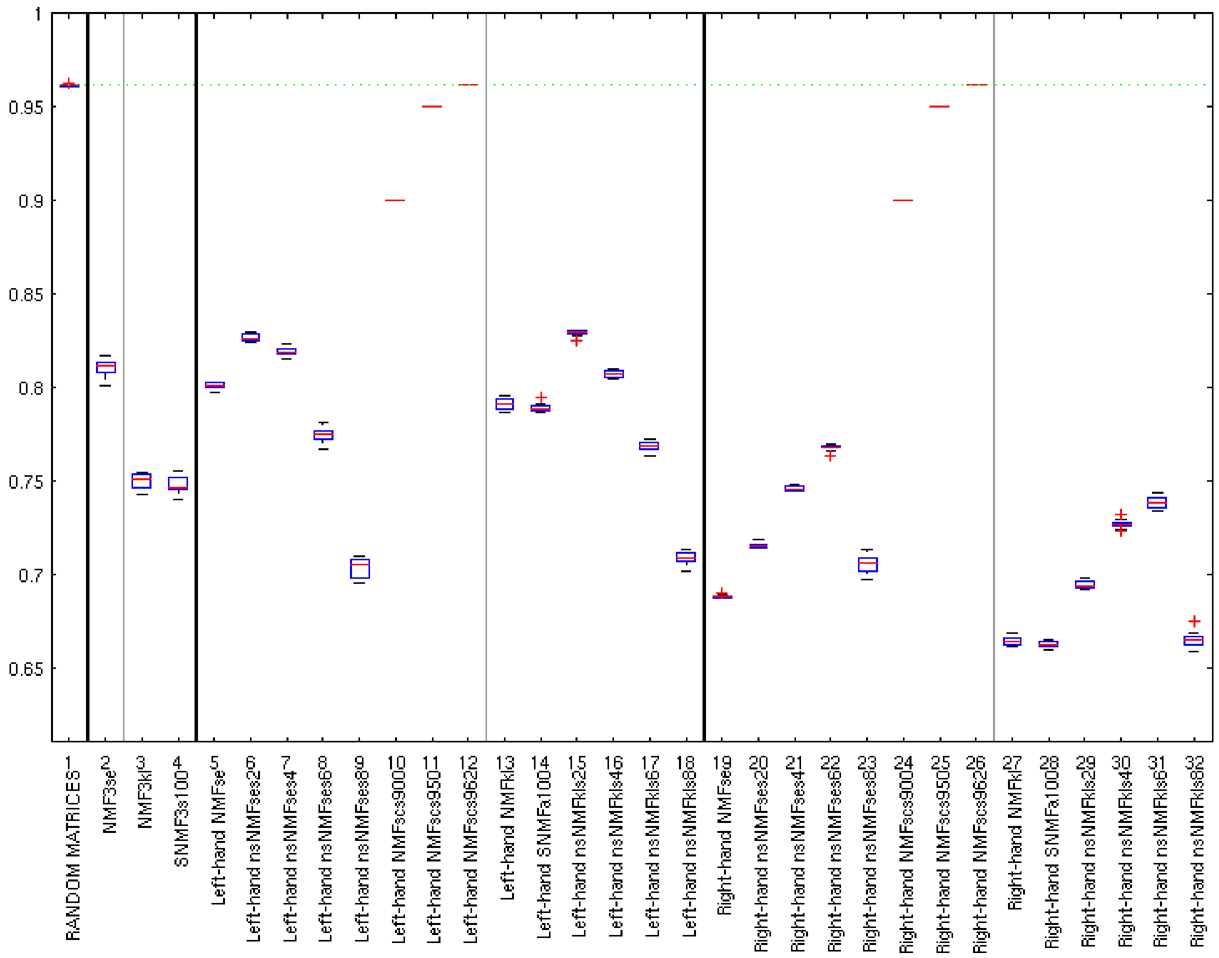


Figure 12: Sparsity of the pathway-reaction matrix P learned from 1500 artificial samples

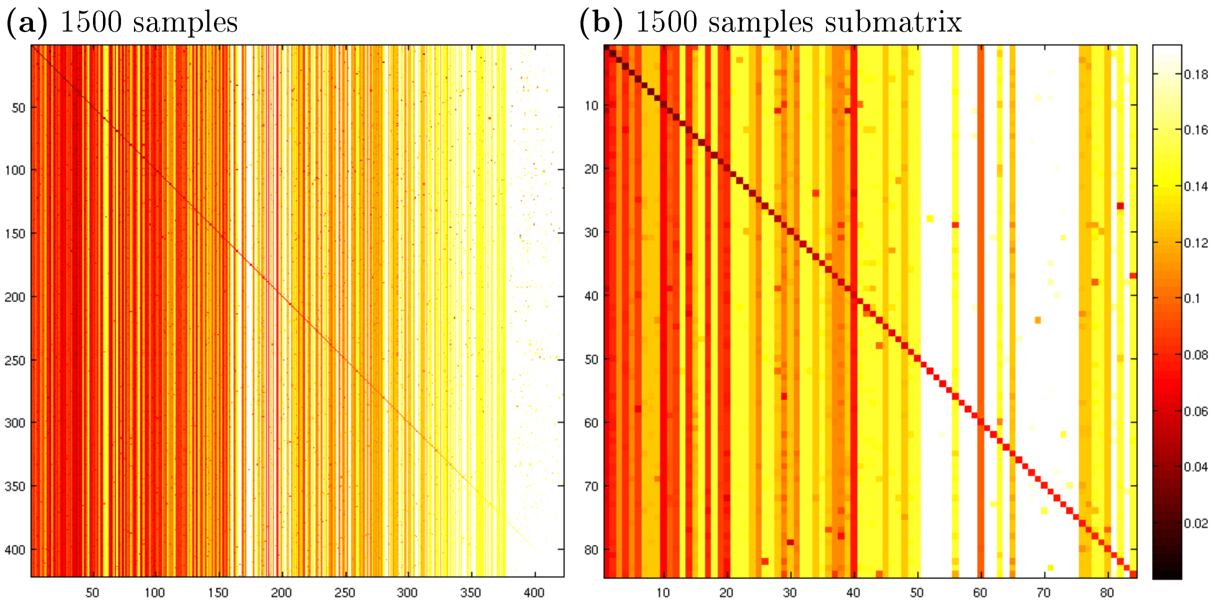


Figure 13: Chessboard plots from one of the chosen factorizations' pathway-reaction matrix after aligning the learned topics with true pathways. The colour of the element (i, j) shows the *squared error* between pathway topic i and true pathway j.

We have further investigated the pathway-reaction matrix resulting from one individual factorization per data set size. For 1500 and 500 samples, we chose factorizations that had most agreed learned pathways. Both of these factorizations resulted from the NMFsc algorithm with sparsity parameter 0.9. On 100 samples, the maximum number of agreed learned pathways, 147, was achieved with NMF-kl. However, we chose the factorization with second most agreed learned pathways, 145, because it was a result from NMFsc with sparsity parameter 0.9, and it only learned two pathways less than the overall best factorization by NMF-kl.

Figure 13 shows chessboard plots of the factorization on 1500 samples and one high-quality submatrix. The plots from 500 and 100 samples look qualitatively similar to Figure 13 (a).

In the chessboard plot, white columns that have a single dark element correspond to true pathways that have low similarity to most pathway topics, but are similar to one. The method has extracted these pathways well. Columns that only have large values correspond to true pathways that we were not able to accurately recover. None of the learned pathway topics is very similar to these true pathways. Fairly dark, evenly coloured columns correspond to true pathways that have high similarity to many pathway topics, and white columns to those that have low similarity to all pathway topics. The method has not learned these pathways accurately.

Figure 13 (b) shows a submatrix of the matrix in 13 (a). It includes only row-column pairs that are constantly aligned to each other when the cost matrices considering Euclidean squared error, KL divergence and absolute error are permuted as described in Section 8.2. There were 113 such pairs, and the 84 shown in Figure 13 (b) make the left upper corner of that matrix.

Measured with absolute error, there were two pathways that were among the ten most accurately learned pathways on all three pathway matrices selected for further analysis. One of them is called creatinine degradation I. It is related to a breakdown product of creatine, which according to MetaCyc is 'used to store and supply energy to muscle cells in vertebrates' but not reabsorbed. The pathway contains 6 enzymes, which is close to the average length in our pathway matrix: 5.6 enzymes. In our artificial data, creatinine degradation I is implemented by 16 different OTUs.

The other well-learned pathway's name is chorismate biosynthesis I, which, *e.g.*, leads to biosynthesis of vitamins E and K. It is 8 enzymes long. Implemented by 89 OTUs, this is a very common pathway in our OTU set, as only 22 pathways are more common, and the average number of implementing OTUs per pathway is 29.

We also examined what pathways the method did not seem to extract. There were five pathways that were among the 40 last true pathways to be aligned by absolute error in all three matrices. This suggests that neither of the three factorizations that were performed on different sizes of data learned these pathways accurately. Four of these pathways are shorter than average, three to five enzymes each, although one of them is a 9 enzymes long superpathway. More importantly, one of these five pathways is only implemented by two OTUs, and the four remaining pathways by only one OTU. This suggests that the method is able to learn more accurately pathways that are implemented by many OTUs. The fact that a high number of OTUs that implement a pathway increases the amount of evidence of that pathway in the samples supports this belief.

In the right-hand approach, the pathways were extracted first as described in Section 4. The resulting sample-pathway matrix was the input for the OTU learning step. The quality of the first factorization strongly affects how well OTUs are predicted. Using only 100 samples, we were not able to recover as many pathways as with higher number of samples, and the number of learned OTUs is also smaller.

Figure 14 shows ROC curves of the matrix P, again from the same factorizations that were previously analyzed in this section and in Section 8.3.1. In addition to the whole matrix P, the figures show ROC curves for two submatrices. The bigger submatrices consist of the intersection of row-column pairs in reordered Euclidean squared error cost matrix and absolute error cost matrix. In the case of the smaller submatrix, the intersection is

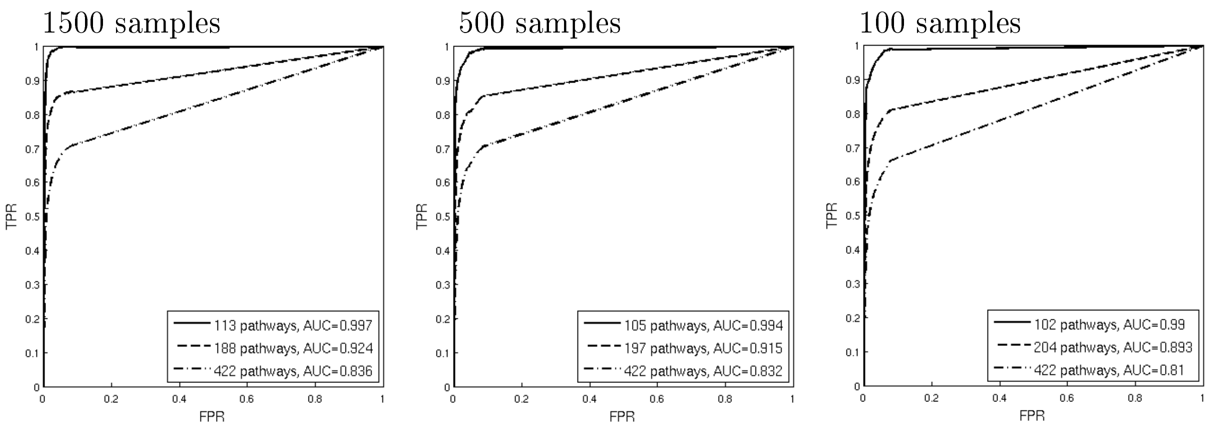


Figure 14: Pathway ROC of the chosen factorization instances on artificial data. AUC value for each curve is shown in the legend.

additionally taken with the KL divergence cost matrix alignments.

Although the number of learned pathways discussed before suggests that many pathways were not recovered well, the ROC curves in Figure 14 show that the learned matrix P estimates all pathways well with AUC higher than 0.8 on all three data sets. The decrease in the AUC values as the number of samples goes down from 1500 to 500 and 100 is in line with the decrease in the number of learned pathways discussed above.

The FPR stays very low while the TPR grows, despite that we know that the underlying pathway matrix is extremely sparse and thus most of the entries are zeros. This means that even though a randomly placed 1 is very likely to be a false positive, in the binarized learned matrix most of the ones are true positives — vast majority of big values in the learned matrix align with ones in the underlying binary matrix.

For comparison, the ROC curve of one of the randomly generated matrices after similar aligning of the cost matrix is shown in Figure 15. The TPR first grows to approximately 0.13. After this, the false positives start to dominate, which means that mainly elements that should be 0 are predicted to be ones when the binarizing threshold α is further decreased. The AUC is only 0.461. This shows that the alignment procedure alone could not make our randomly generated matrices look like good estimators.

In Figure 14, the smallest submatrices cover around one quarter of all pathways. The method learned these over 100 pathways very accurately. Their TPR reaches almost the value 1 with only a low FPR. This means that with some threshold, the learned matrix is binarized so that almost every 1 of the underlying matrix is found while only a small amount of zeros are predicted to be ones. This means that these submatrices have correctly predicted almost all reactions that are present in the pathways in question, while including few reactions that are not present in the underlying pathway.

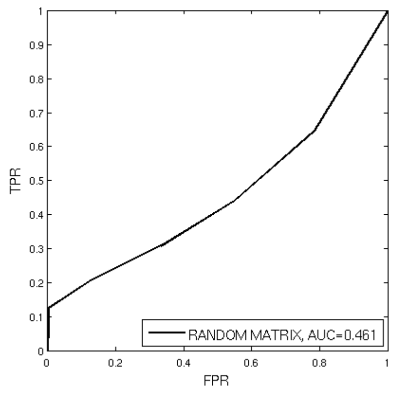


Figure 15: ROC curve of one of the randomly generated pathway-reaction matrices

Note that the individual factorizations that we chose for closer investigation were selected as a compromise between the performance on extracting OTUs and pathways. These matrices do not represent the overall best result of extracting pathways.

8.3.3 OTU-pathway relationships from artificial data

Only knowing what underlying OTUs the rows of H and what underlying pathways the columns of H correspond to allows investigating the OTU-pathway matrix H closely. For this reason, we validate the learned matrix H from the same three individual factorizations that are analyzed in Sections 8.3.1 and 8.3.2. They were chosen to represent our results well, but a compromise between the learned OTUs and pathways was made.

OTU topics of the matrix W and pathway topics of the matrix P are already aligned before examining how OTUs and pathways were predicted. In the matrix H, we permute the rows and columns to the corresponding order: each column k of W correspongs to row k of H, and similarly each row l of P corresponds to column l of H. Now that also H is permuted like this, the matrix product WHP has exactly the same value as before any of the matrices was permuted. Hence, the permuted OTU-pathway matrix H tells how our NMF method predicted the relationships between the OTU and pathway topics. When validating H, we compare the predicted OTU-pathway distributions with the OTU-pathway distributions of the same underlying OTUs and pathways to which the topics were aligned in Sections 8.3.1 and 8.3.2.

Figure 16 shows ROC curves for the OTU-pathway matrix of the selected factorizations after normalizing the matrices row-wise. All curves show that the relationships between OTUs and pathways were predicted better than what a random guess would give. This suggests that it was not by chance that the method extracted OTU-like and pathway-like features, but that it was able to learn the latent variables of the data.

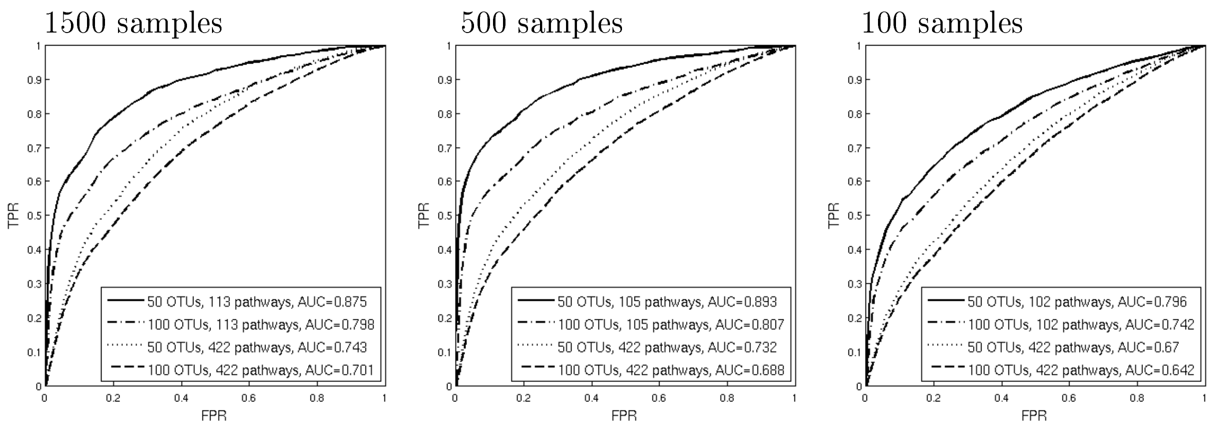


Figure 16: ROC curves of an OTU-pathway matrix and selected submatrices

When only subsets of OTUs and pathways that were most accurately learned are considered, the AUC is higher. When the OTUs and pathways were accurately learned, their relationships were well-predicted as well.

When all 422 pathways are taken into account, we also consider the pathway topics that do not correspond to any real pathway, as discussed in the previous subsection. Hence, the errouneously predicted pathways affect the OTU-pathway matrix, and the resulting AUC values are not as high as when only the accurately learned pathways are considered.

When only the around 100 pathways that all three measures align similarly are taken into account, we see that the FPR stays very low as the TPR grows to 0.3 or 0.4. The biggest values of these submatrices are in the same indices than ones in the underlying binary matrix. This phenomenon is not as strong on all 422 pathways.

Considering 50 of the most accurately learned OTUs raises the ROC curves in all cases, especially when also the less well learned pathways are ignored.

As the number of samples is decreased from 1500 to 500, the AUC values stay about the same. With only 100 samples, the values drop more. These results are in line with how well OTUs and pathways were extracted from these different sizes of data sets.

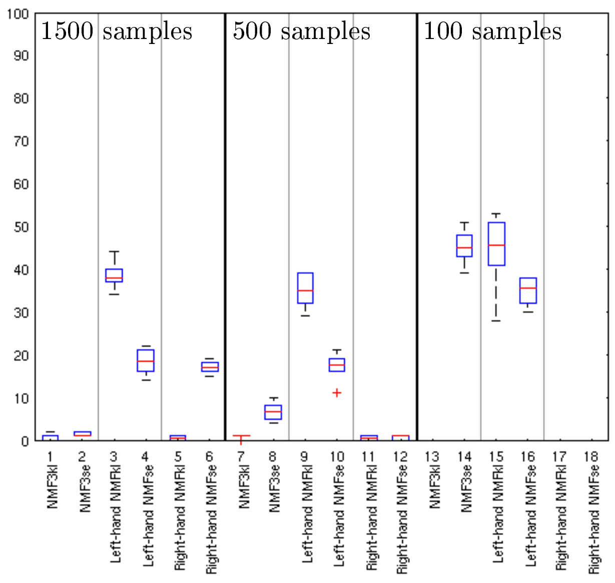


Figure 17: Number of agreed OTUs learned from 1500, 500 and 100 simulated samples

8.4 Experiments on simulated data

Based on the results from Task 1, we chose a few algorithms for the second task. As some of the Euclidean squared error-based methods were able to extract more pathways than any KL divergence-based variant, we only used Euclidean squared error-based ones for learning pathways from the simulated data.

The sampling process described in Section 7 introduces noise and we do not expect an exact factorization with the inner dimensions that we used to exist for our simulated data. Still, the initial distributions define the latent features we hope to recover with our methods. Therefore, we use similar evaluation methods as in the first task, treating the initial distributions as ground truth.

8.4.1 OTUs from simulated data

We found that on simulated data some of the algorithms learn too sparse sample-OTU matrices. On 100 samples they become almost like scaled identity matrices. Most rows and columns only have one or two relatively extremely large elements. Other elements are zero or almost zero. When the columns are normalized to attain OTU topics, the small elements underflow and become zeros. The learned mapping from samples to OTUs being nearly a one-to-one mapping means these learned sample-OTU matrices only assign one or a few OTUs to each sample.

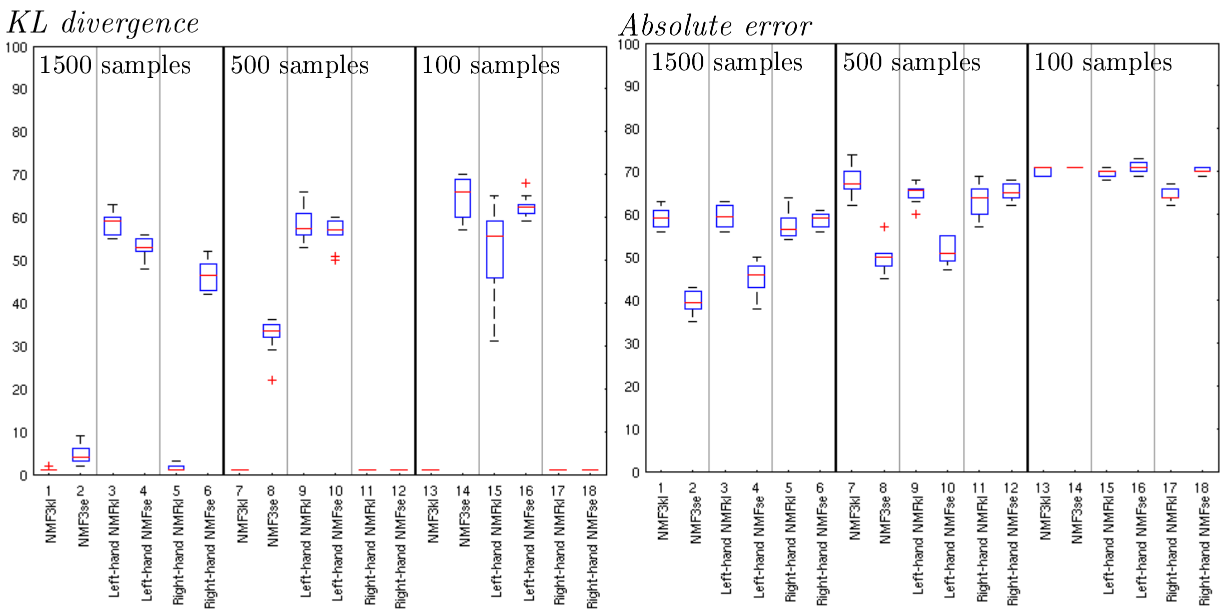


Figure 18: Number of OTUs learned from 1500, 500 and 100 simulated samples measured using one metric at a time

Figures 17 and 18 show that the number of learned OTUs from simulated data is constantly below 70, which is lower than from artificial data. Interestingly, the numbers increase as the data set size decreases. Every underlying sample distribution contains every OTU, so, the true OTUs do not have any zero elements. Unfortunately, that means the KL divergence to an OTU topic that contains at least one zero is undefined. This shows in Figures 17 and 18 where the number of learned OTUs is 1 or even missing when KL divergence is used.

Similarly as with artificial data, we chose one individual factorization per data set size for further analysis. For easier comparison, we chose the left-hand factorization as with artificial data. The chosen matrices serve as the input data for the second step of the left-hand factorization, *i.e.* the ones with most agreed learned OTUs. These matrices were produced by the NMF-kl algorithm.

Figure 19 shows chessboard plots corresponding to Figure 9 on Page 52, apart from that the submatrix is different. The contrast between the diagonal and off-diagonal elements of the submatrix tells that as with artificial data, selecting the agreed alignings results in subsets where the OTUs are visibly better distinguished from each other. The scale shows that the divergences of the results on simulated data are greater in all factorizations compared to artificial data.

Whereas the contrast grows between the elements of matrices shown in Figure 9 when the number of samples decreases, in Figure 19 the matrices become smoother. This means the similarity between a given learned topic and all the true OTUs is more at the same level, while on artificial data the OTU topics were very similar to only a few true OTUs and different from the others. We interpret that the method has not been able to distinguish the underlying OTUs as well as on artificial data. The low numbers of learned OTUs discussed above supports this.

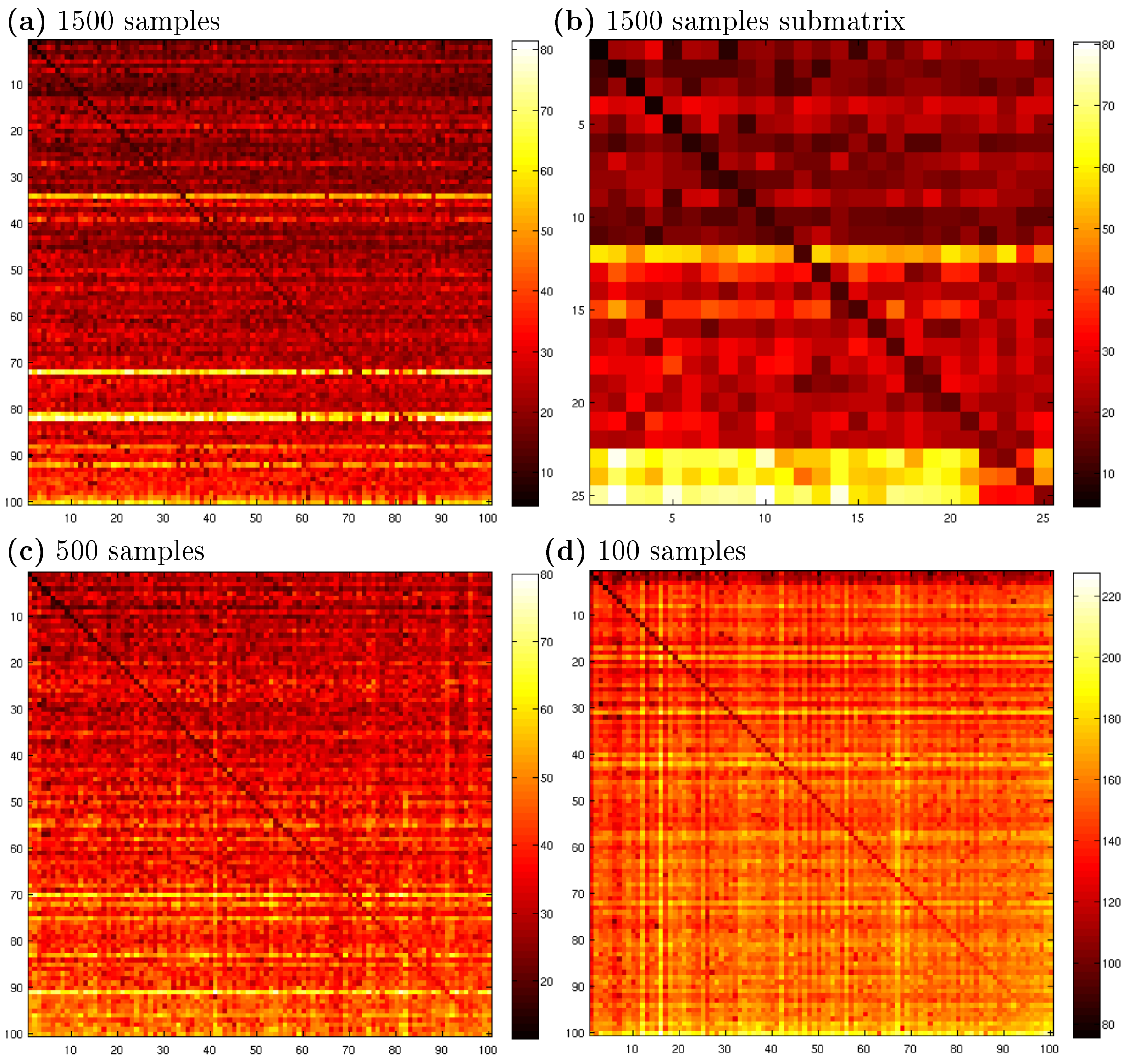


Figure 19: Chessboard plot from selected sample-otu matrices after aligning the learned topics with true OTUs. The colour of element (i, j) shows the *KL divergence* from true OTU j to OTU topic i. Higher divergences than 10 are displayed using white colour.

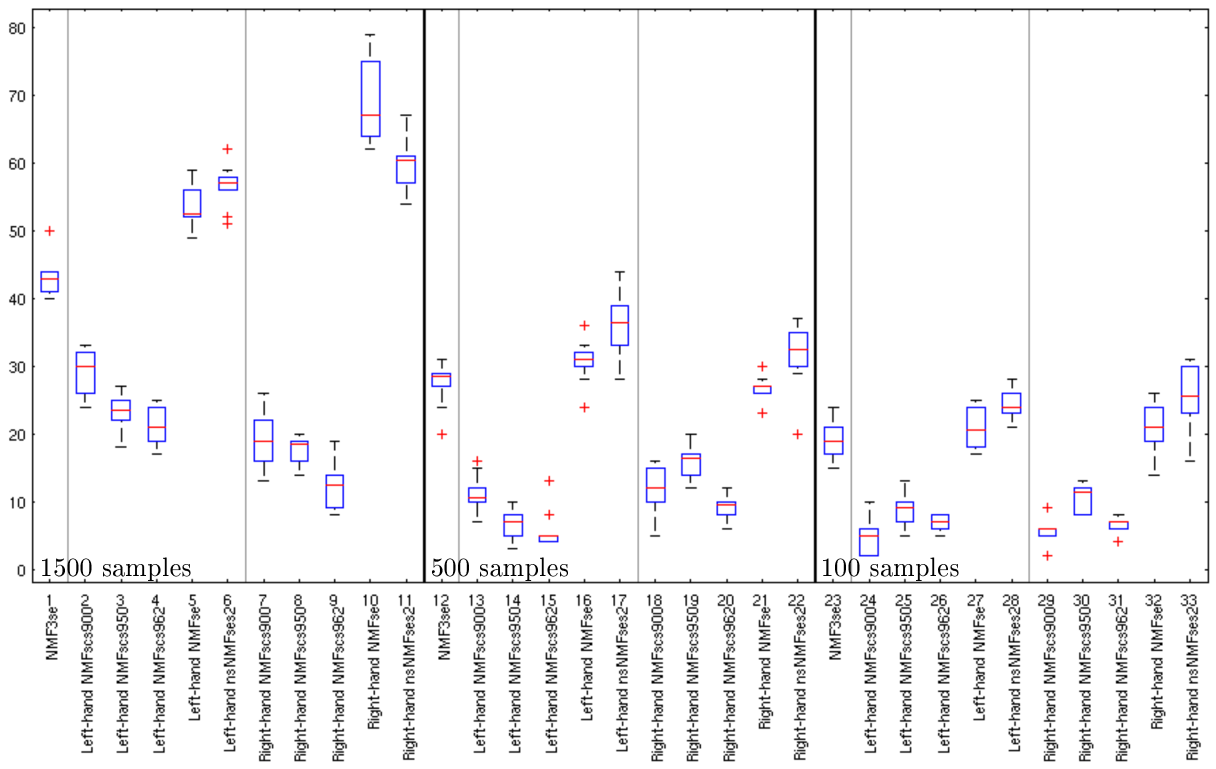


Figure 20: Number of agreed pathways each of the algorithms learned from 1500, 500 and 100 simulated samples

8.4.2 Pathways from simulated data

The methods were not able to learn as many pathways as on artificial data, which is seen by comparison of Figures 20 and 11. The drop is especially visible in the number of agreed learned pathways, and when KL divergence alone is used. In general, left-hand factorizations performed visibly better on the artificial data, but this trend cannot be seen in Figures 20 or 21. On the contrary, on simulated data most pathways were extracted by NMF-se in the right-hand setup.

As on artificial data, most methods' performance was better on higher numbers of samples. The decrease that happens when the data set size is dropped from 1500 to 500 is more dramatic in the case of NMF-se and nsNMF-se in both the left-hand and the right-hand factorizations.

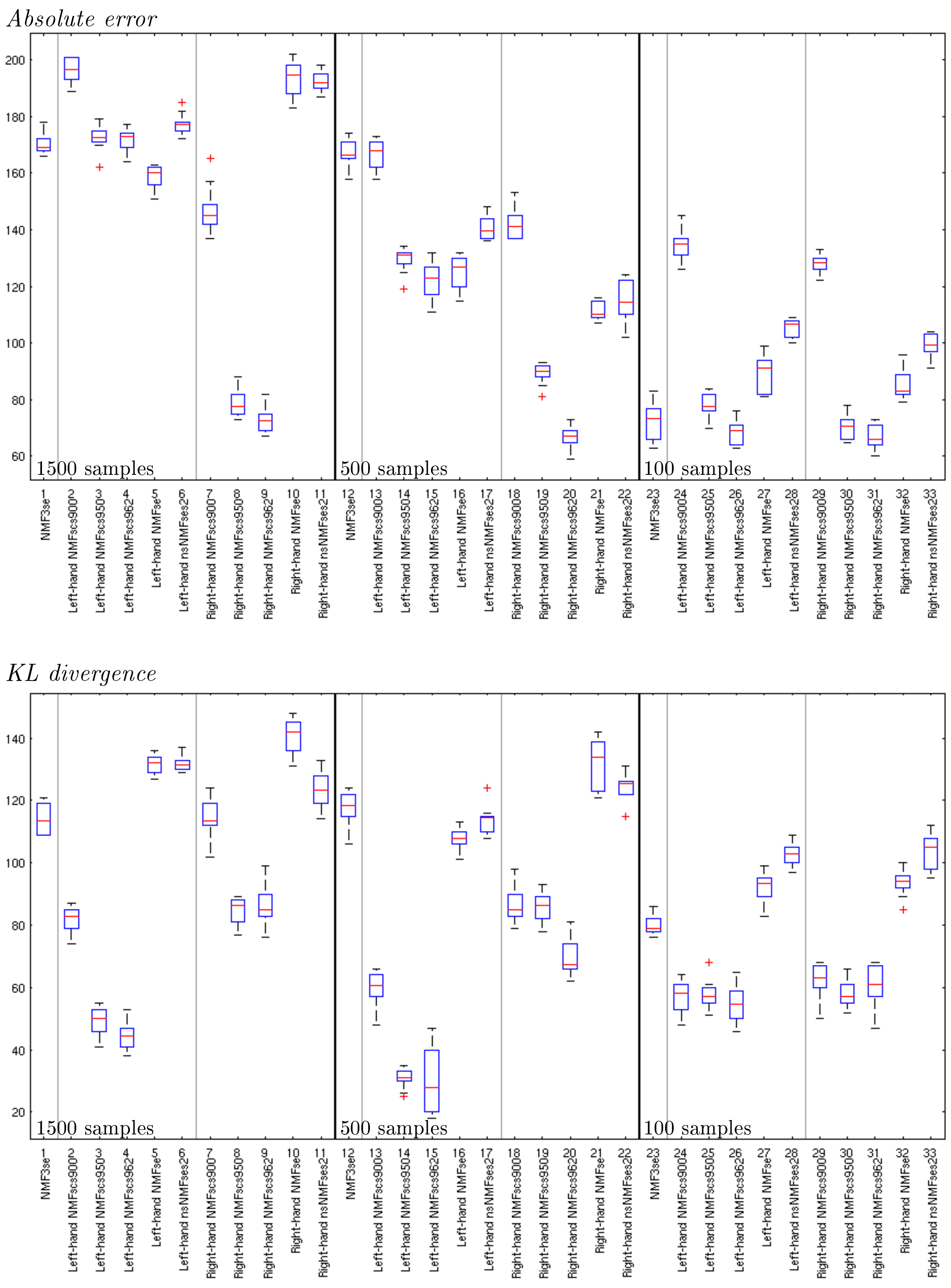


Figure 21: Number of pathways each of the algorithms learned from 1500, 500 and 100 simulated samples measured using either KL divergence or absolute error

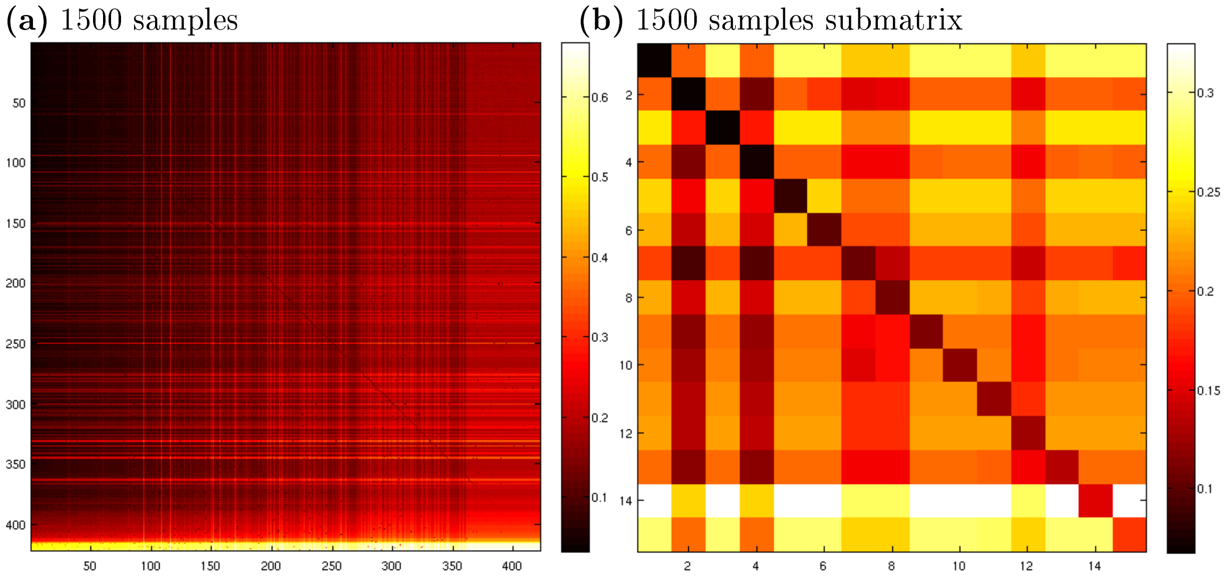


Figure 22: Chessboard plot from the chosen pathway-reaction matrix on 1500 samples and a high-quality submatrix after aligning the learned topics with true pathways. The colour of the element (i, j) shows the *Euclidean squared error* between true pathway j and pathway topic i.

We again chose three individual factorizations for further analysis. For each data set size, we selected the factorization with most agreed learned pathways. These were achieved with the nsNMF-se algorithm. On 1500 and 100 samples, the sparsity parameter was 0.4, and on 500 samples it was 0.2. Note that the algorithm is different from the one that resulted in the matrices we analyzed on artificial data.

Figure 22 (a) shows the chessboard plot of the Euclidean squared errors between pathway topics and real pathways. Corresponding plots for both 500 and 100 samples are qualitatively similar. On artificial data, the chessboard plots shown in Figure 13 on page 57, show a visible contrast between different columns. Instead, in the matrix in Figure 22, the values are almost symmetrically spread.

The submatrix shown in Figure 22 (b) contains elements that KL divergence, absolute error and Euclidean squared error align similarly. It shows that considering this kind of agreed alignment selects pathways that the algorithm has been able to separate more clearly than on average. Note that there were 113 such pathways extracted from the artificial data, while the number is only 15 on simulated data. This suggests that the method was able to learn pathways more accurately from artificial data, although the results cannot be directly compared because a different variant of the algorithm was used.

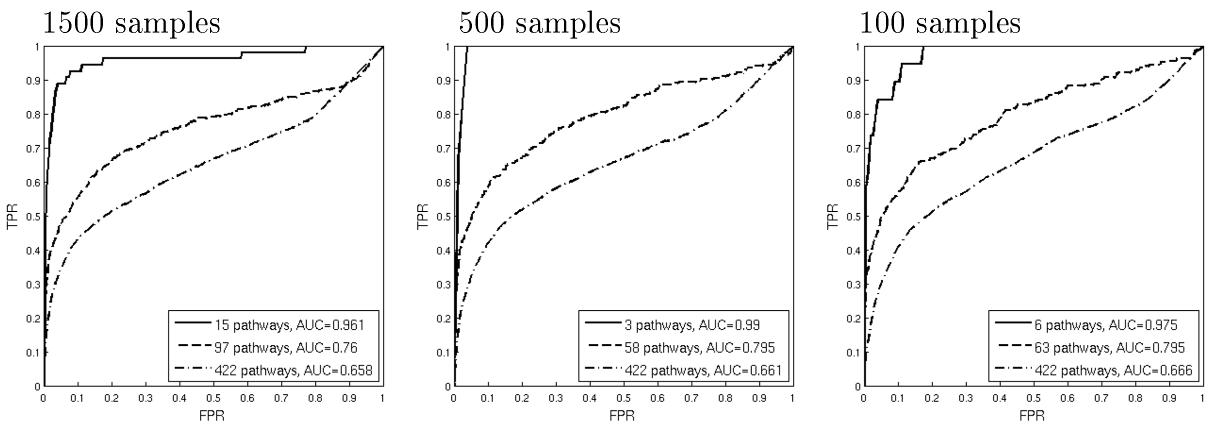


Figure 23: Pathway ROC of the chosen factorization instances

Figure 23 shows ROC curves for the three selected matrices. The submatrices were chosen similarly as on artificial data. Note that the submatrices are smaller than on artificial data.

The ROC curves produced from artificial data shown in Figure 14 show that all pathways were learned well, and more than 100 pathways very accurately. Figure 23 suggests that the matrix did not estimate all pathways as well. The AUC values are not as high as on artificial data. The size of the bigger submatrix is smaller than the size of the smallest submatrix in Figure 14, but the AUC is much lower.

On artificial data, the number of false positives increased very slowly as the threshold was moved down from 1. When the estimate put a large weight on the element (l, j), the reaction j did in fact belong to pathway l. On simulated data, the large values of the learned matrices are not as well aligned with the ones in the underlying binary matrix. On the whole matrix and the bigger submatrix, the FPR starts growing significantly earlier than on artificial data. Only a few pathways that the smaller submatrices consist of were estimated well.

8.4.3 OTU-pathway relationships from simulated data

Figure 24 shows ROC curves for the OTU-pathway matrix of the selected factorizations. When all pathways are included, the results follow the diagonal which shows that the prediction resembles a random guess. On 1500 and 100 samples, taking the submatrices improves the estimate. On 500 samples, the estimate unfortunately goes below the diagonal.

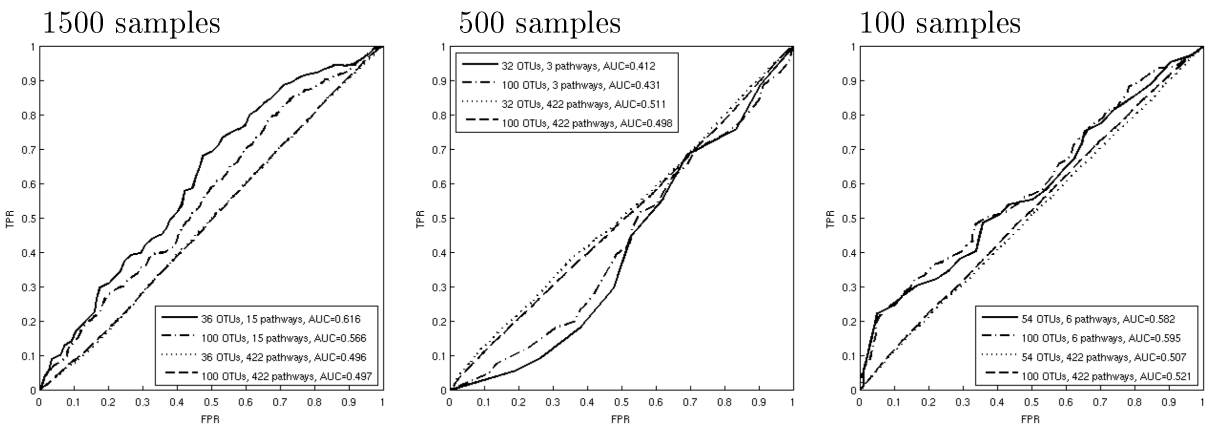


Figure 24: ROC curves of an OTU-pathway matrix and selected submatrices

9 Conclusion

We found that the NMF methods were able to extract most OTUs from the artificial data. The ROC curves in Figure 14 show that the method was able to predict most pathways well, and almost one half of the pathways very accurately. The relationships between around 100 most accurately learned pathways and all OTUs are well predicted with AUC around 0.8 when 1500 or 500 samples were used. The AUC grows to nearly 0.9 when we further discard one half of the OTUs.

On artificial data, the quality of all three learned matrices was highest on 1500 samples but in most cases almost as high on 500 samples. With only 100 samples available, the results were still surprisingly good despite that it is very difficult to predict many features from this few samples.

In general, KL divergence-based algorithms extracted more OTUs than Euclidean squared error-based ones. In contrast, the performance of se-variants on predicting pathways was better than KL divergence-based methods. We expect this due to the fact that the OTUs were defined by sampling from Dirichlet distribution while the nature of our pathway data is binary.

Our novel three-matrix variants NMF3-kl and SNMF3 extracted more OTUs than any other variants. Surprisingly, the highest numbers of pathways was extracted by left-hand factorizations in which the OTUs are extracted first. Many algorithms were able to benefit from their sparsity constraints although the soft constrained algorithms were not able to produce as extremely sparse matrices as our underlying pathway-reaction matrix.

All NMF methods were able to recover more features of our interest from the artificial data than from simulated data. Especially the quality of the OTU-pathway matrix was very different on these two kinds of data. Changing the number of samples between 100 and 1500 did not affect the results of the factorization as much as changing from the artificial to simulated data.

The most fundamental difference between artificial and simulated data is the sampling depth. An artificial sample contains exactly the right amount of evidence from all reactions present in the sample. Besides of this data being noise-free, no information of the reactions is lost. We conclude that the sampling depth of 1000 was not sufficient for any of the methods to extract the OTU-pathway relationships.

10 Future Work

It would be interesting to study how the results change as the sampling depth is increased from 1000 to other finite numbers. Varying the number of latent variables and the dimensions of the input data could also be worth studying.

As with many gradient descent-based methods, initialization can strongly affect which local minimum is found. In addition, starting near an optimum may speed up the convergence. Using other existing techniques as a preprocessing phase and initializing one or more of the matrices W, H and P with OTU and pathway predictions resulting from these other methods would be interesting.

The sparsity constrained NMF variants we investigated were based on L^1 or L^2 norms, or relied on the intuitive meaning of sparsity. Because the pathway information has a binary nature, we are interested in a variant that could explicitly control the number of zeros in the matrix P.

Other interesting methods may include matrix factorization techniques with binary output matrices. Especially a method that could factorize a real-valued matrix into a real-valued OTU matrix and a binary pathway matrix would conveniently fit the problem setting. An idea of this kind of factorization method was suggested by Slawski *et al* [41].

References

[1] Minoru Kanehisa and Susumu Goto. *KEGG: Kyoto Encyclopedia of Genes and Genomes*. Nucleic acids research 28.1 pp 27-30, 2000.

[2] UniProt Consortium. *The Universal Protein Resource (UniProt)*. Nucleic acids research 36 suppl 1 pp D190-D195, 2008.

[3] Caspi *et al*. *The MetaCyc Database of Metabolic Pathways and Enzymes and the BioCyc Collection of Pathway/Genome Databases*. Nucleic Acids Research, 2012.

[4] R.D. Finn, A. Bateman, J. Clements, P. Coggill, R.Y. Eberhardt, S.R. Eddy, A. Heger, K. Hetherington, L. Holm, J. Mistry, E.L.L. Sonnhammer, J. Tate, M. Punta. *The Pfam Protein Families Database*. Nucleic Acids Research Database Issue 42 pp D222-D230, 2014.

[5] Douglas B. Rusch *et al*. *The Sorcerer II Global Ocean Sampling Expedition: Northwest Atlantic Through Eastern Tropical Pacific*. PLoS biology 5.3: e77, 2007.

[6] Jack A. Gilbert *et al*. *Meeting report: The Terabase Metagenomics Workshop and the Vision of an Earth Microbiome Project*. Standards in genomic sciences 3.3, 243.2010

[7] Jane Peterson, Susan Garges, Maria Giovanni, Pamela McInnes, Lu Wang, Jeffery A. Schloss, Vivien Bonazzi *et al*. *The NIH Human Microbiome Project*. Genome research 19, no. 12 pp 2317-2323, 2009.

[8] Philip Hugenholtz, Brett M. Goebel, Norman R. Pace. *Impact of Culture-independent Studies on the Emerging Phylogenetic View of Bacterial Diversity*. Journal of bacteriology 180.18 pp 4765-4774, 1998.

[9] Rudolf I. Amann, Wolfgang Ludwig, Karl-Heinz Schleifer. *Phylogenetic Identification and in Situ Detection of Individual Microbial Cells Without Cultivation*. Microbiological reviews 59.1 pp 143-169, 1995.

[10] Michelle R. Rondon *et al*. *Cloning the Soil Metagenome: a Strategy for Accessing the Genetic and Functional Diversity of Uncultured Microorganisms*. Applied and environmental microbiology 66.6 pp 2541-2547, 2000.

[11] Venter, J. Craig, *et al*. *Environmental Genome Shotgun Sequencing of the Sargasso Sea*. Science 304.5667 pp 66-74, 2004.

[12] Qin, Junjie, *et al*. *A Human Gut Microbial Gene Catalogue Established by Metagenomic Sequencing*. Nature 464.7285 pp 59-65, 2010.

[13] Jiajia Ni, Qingyun Yan, Yuhe Yu. *How Much Metagenomic Sequencing is Enough to Achieve a Given Goal?*. Scientific reports 3, 2013.

[14] Philip Hugenholtz1, Gene W. Tyson. *News and Views Q&A Microbiology: Metagenomics*. Nature 455 pp 481-483, 2008.

[15] William R. Pearson and David J. Lipman. *Improved Tools for Biological Sequence Comparison*. Proceedings of the National Academy of Sciences 85.8 pp 2444-2448, 1988.

[16] Stephen F. Altschul *et al*. *Basic Local Alignment Search Tool*. Journal of molecular biology 215.3 pp 403-410, 1990.

[17] Daniel H. Huson *et al*. *MEGAN Analysis of Metagenomic Data*. Genome research 17.3 pp 377-386, 2007.

[18] Xingpeng Jiang, Joshua S. Weitz, Jonathan Dushoff. *A Non-Negative Matrix Factorization Framework for Identifying Modular Patterns in Metagenomic Profile Data*. Journal of Mathematical Biology, vol. 64, no. 4 pp 697-711, 2012.

[19] Arumugam, Manimozhiyan, *et al*. *Enterotypes of the Human Gut Microbiome*. Nature 473.7346 pp 174-180, 2011.

[20] Xingpeng Jiang, Joshua S. Weitz and Jonathan Dushoff. *A Non-negative Matrix Factorization Framework for Identifying Modular Patterns in Metagenomic Profile Data*. Journal of Mathematical Biology 64.4 pp 697-711, 2012.

[21] Xingpeng Jiang, Morgan G. I. Langille, Russell Y. Neches, Marie Elliot, Simon A. Levin, Jonathan A. Eisen, Joshua S. Weitz, Jonathan Dushoff. *Functional Biogeography of Ocean Microbes Revealed through Non-Negative Matrix Factorization*. PLoS ONE 7(9): e43866, 2012.

[22] Teeling, Hanno, *et al*. *TETRA: a Web-service and a Stand-alone Program for the Analysis and Comparison of Tetranucleotide Usage Patterns in DNA Sequences*. BMC bioinformatics 5.1 pp 163, 2004.

[23] Hao Zheng and Hongwei Wu. *Short Prokaryotic DNA Fragment Binning Using a Hierarchical Classifier Based on Linear Discriminant Analysis and Principal Component Analysis*. Journal of Bioinformatics and Computational Biology 8.06 pp 995-1011, 2010.

[24] Mohammed, Monzoorul Haque, *et al*. *SPHINX — an Algorithm for Taxonomic Binning of Metagenomic Sequences*. Bioinformatics 27.1 pp 22-30, 2011.

[25] Christophe H. Schilling, David Letscher and Bernhard Ø. Palsson. *Theory for the Systemic Definition of Metabolic Pathways and Their Use in Interpreting Metabolic Function from a Pathway-oriented Perspective.* Journal of theoretical biology 203.3 pp 229-248, 2000.

[26] Stefan Schuster, Thomas Dandekar and David A. Fell. *Detection of Elementary Flux Modes in Biochemical Networks: a Promising Tool for Pathway Analysis and Metabolic Engineering.* Trends in Biotechnology 17.2 pp 53-60, 1999.

[27] Darvas, Ferenc. *Predicting Metabolic Pathways by Logic Programming.* Journal of Molecular Graphics 6.2 pp 80-86, 1988.

[28] Hong-Wu Ma *et al. Decomposition of Metabolic Network into Functional Modules Based on the Global Connectivity Structure of Reaction Graph.* Bioinformatics 20.12 pp 1870-1876, 2004.

[29] Jason A. Papin *et al. Metabolic Pathways in the Post-genome Era.* Trends in Biochemical Sciences 28.5 pp 250-258, 2003.

[30] Daniel C. McShan, S. Rao and Imran Shah. *PathMiner: Predicting Metabolic Pathways by Heuristic Search.* Bioinformatics 19.13 pp 1692-1698, 2003.

[31] Sanjeev Arora, Rong Ge, Ravi Kannan, Ankur Moitra. *Computing a nonnegative matrix factorization – provably.* Proceedings of the forty-fourth annual ACM symposium on Theory of computing pp 145-162, 2012.

[32] Stephen A. Vavasis. *On the complexity of nonnegative matrix factorization.* SIAM Journal on Optimization, 20.3 pp 1364-1377, 2009.

[33] P. Paatero, U. Tapper. *Positive Matrix Factorization: A Non-negative Factor Model with Optimal Utilization of Error Estimates of Data Values.* Environmetrics pp 111-126, 1994.

[34] Daniel D. Lee, H. Sebastian Seung. *Learning the Parts of Objects by Non-Negative Matrix Factorization.* Nature 401 pp 788-791, 1999.

[35] Daniel D. Lee, H. Sebastian Seung. *Algorithms for Non-Negative Matrix Factorization.* Advances in Neural Information Processing 13. Proc. NIPS 2000, MIT Press, 2001.

[36] Patrik O. Hoyer. *Non-Negative Matrix Factorization with Sparseness Constraints.* The Journal of Machine Learning Research, vol. 5 pp 1457-1469, December 2004.

[37] Alberto Pascual-Montano, J.M. Carazo, Kieko Kochi, Dietrich Lehmann, Roberto D. Pascual-Marqui. *Nonsmooth Nonnegative Matrix Factorization (nsNMF).* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 3, March 2006.

[38] Weixiang Liu, Nanning Zheng, Xiaofeng Lu. *Non-negative Matrix Factorization for Visual Coding.* Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP'03). 2003 IEEE International Conference on, vol. 3 pp III-293, April 2003.

[39] Stan Z. Li, XinWen Hou, HongJiang Zhang, QianSheng Cheng. *Learning Spatially Localized, Parts-based Representation.* Computer Vision and Pattern Recognition, 2001. (CVPR 2001). Proceedings of the 2001 IEEE Computer Society Conference on, vol. 1 pp I-207,I-212, 2001.

[40] Chris Ding *et al*. *Orthogonal nonnegative matrix t-factorizations for clustering.* Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, 2006.

[41] Martin Slawski, Matthias Hein, Pavlo Lutsik. *Matrix factorization with binary components.* Advances in Neural Information Processing Systems, 2013.

[42] http://www.mathworks.se/help/stats/boxplot.html, visited June 18, 2014.