ERASMUS UNIVERSITY ROTTERDAM
Erasmus School of Economics

A Novel ℓ_0-Sparse Nonnegative Matrix Factorization Algorithm (ℓ_0-SNMF) for Community Detection

An Approach Using Iterative Majorization

Master Thesis Business Analytics and Quantitative Marketing

Author: Dana de Leeuw
Student ID number: 416556dl

Supervisor: Prof. Dr. P.J.F. Groenen
Second assessor: Prof. Dr. Ş.I. Birbil

Abstract

In this work, we proposed a novel ℓ_0-Sparse Nonnegative Matrix Factorization (ℓ_0-SNMF) algorithm for the community detection task. It improves upon the previous baseline for NMF algorithms with a ℓ_0-(pseudo)norm sparsity constraint, which was set by Peharz and Pernkopf (2012). The novel algorithm is also capable of enforcing the ℓ_0-constraint for alternative NMF loss functions, which is not possible with the previous baseline. Iterative majorization was used to obtain a final updating expression. The updating rule decreases the loss function of NMF as much as possible with the best subset, whilst enforcing ℓ_0-sparsity. We have implemented the novel ℓ_0-SNMF algorithm on one synthetic and seven real-world network datasets, all widely used in academic research. Additionally, we tested its robustness through increasing contamination levels. The results demonstrate that the novel algorithm performs substantially better than the ℓ_0-sparse baseline when fixing the number of nonzero elements to one for sparse networks. It attains a higher Normalized Mutual Information under contamination and lower computation time with a slightly slower convergence. For the unconstrained ℓ_0-SNMF algorithm we conclude that performance is at least comparable to classic NMF in terms of NMI and robustness, whilst it has a lower computation time. We also observe that ℓ_0-constrained algorithms have in general a more stable performance than unconstrained ones for unknown number of communities and noise.

Keywords: Nonnegative Matrix Factorization, NMF, ℓ_0-sparse, Iterative Majorization, Sparse NMF, Community Detection

January 20, 2020

Contents

1 Introduction

Networks science is a relatively new field of study, which can be dated back to the beginnings of the 20$^{\text{th}}$ century. Nonetheless, this field of study has gained large popularity due to its extensive applicability to different types of networks; from informational networks which include for example the World Wide Web, to social networks such as Facebook relations, or even biological networks such as the ecosystem of the rain forest. All real-world networks have one thing in common; the presence of a community structure. It is by identifying and analyzing this structure that novel insights can be gained. To do this, networks are modeled as graphs consisting of nodes/vertices, which are interlinked by edges.

One such analysis is the grouping of elements within a network into a community, called community detection. In many cases, such communities represent functional units in the network. By obtaining different communities, information can be gathered on how the network functions and how the topology affects the individual nodes. As properties within a community can be considerably different when compared to the network as a whole, it is important to perform community detection. It is by ignoring these community-specific properties that an opportunity is missed to understand and possibly capitalize on these differences. Additionally, by performing community detection we can determine the probability of a vertex being part of a certain community. As such, it is possible to identify false links within a network. It is for these reasons that algorithms with the ability to solve the community detection task have become popular (Cannistraci et al., 2013; Girvan and Newman, 2002).

No formal definition exists of a community, as the underlying drivers per community vary greatly depending on the application. However, the most accepted definition is that a community is a set of nodes densely connected between each other and simultaneously less connected to other nodes in the network (Fortunato, 2010). The community detection task aims to determine such modules so that the underlying structure can be derived by topological information of the network or similarity measures.

Numerous different methods have been introduced for solving community detection tasks. Following Javed et al. (2018) which builds upon the extensive study of Fortunato (2010), these methods can be divided based on their structural properties into disjoint or overlapping community detection algorithms. As these names suggest, the communities detected by the former are completely separate meaning that nodes can only belong to one community, whereas for the latter a node can still maintain links to other communities. As most real-world networks have overlapping communities due to their complexity and interconnectedness, Fortunato and Hric (2016) concludes that overlapping community detection algorithms are better suited to represent real-world problems. Additionally, overlapping community detection algorithms are capable of solving both disjoint and overlapping tasks.

For all community detection algorithms, nonnegative matrix factorization (NMF) approaches have been continuously popular. Mainly due to their ability to effectively determine the underlying communities, while simultaneously being easily interpretable (Wang et al., 2011). NMF decomposes a given matrix into two nonnegative matrices to determine the latent features of a structure (Lee and Seung, 1999). It is capable of handling all different network types: directed/undirected, weighted/unweighted and disjoint/overlapping. Moreover, the loss function can be altered to become more robust (Kong et al., 2011) or to

include regularization terms, such as graph regularization (Cai et al., 2010; Wu et al., 2018) or sparsity regularization (Peharz and Pernkopf, 2012).

Within sparsity regularized NMF algorithms, it might be of interest to let nodes only belong to a restricted number of communities, as this can result in a sparse yet accurate representation of the network. Little research has focused so far on the use of this ℓ_0-norm as a sparsity constraint for NMF, even though sparsity is defined by this norm. Peharz and Pernkopf (2012) introduced the current baseline algorithm for ℓ_0-sparse NMF. Nonetheless, this algorithm is based on a combination of nonnegative least squares and orthogonal matching pursuit. Due to this, it is computationally expensive since it considers all column combinations. Moreover, it can solely solve the classic NMF problem and not any alternative loss function.

Inspired by Xiong (2013), who shows that for high-dimensional linear regression it is possible to have an efficient algorithm to solve the ℓ_0-(pseudo)norm constrained regression problem, we derive an expression which is capable of enforcing ℓ_0-sparsity on classic NMF, any type of weighted NMF (which includes all robust versions of NMF), and graph regularized NMF. Xiong (2013) uses the expectation maximization (EM) algorithm, which is a special case of the iterative majorization (IM) algorithm. In this work, we use IM to obtain a general updating rule for the novel ℓ_0-sparse NMF algorithm. The contribution of this work is, therefore, as follows:

- The proposed model is capable of not only enforcing ℓ_0-sparsity on classic NMF, but all alternative NMF loss functions.
- Through the use of iterative majorization, we get a more efficient updating rule, which is guaranteed to converge to at least a local minimum.

It must be noted that throughout this work we assume that the number of communities K is known, as we focus on deriving a novel ℓ_0-sparse loss function for NMF. The method by Bordenave et al. (2015) is often used for finding this number. Additionally, since it is already known that NMF is capable of solving both disjoint and overlapping community detection tasks, we focus on disjoint community detection, as there is a larger availability of labeled networks for measuring the performance of algorithms.

The remainder of this paper is organized as follows. In Section 2 we introduce nonnegative matrix factorization, including a discussion of the main algorithms for solving NMF. Afterwards, Section 3 describes the community detection problem and presents different variations of the NMF loss function to solve the community detection task. Section 4 gives a detailed explanation of the newly proposed ℓ_0-SNMF method. This method is compared to the baseline ℓ_0-sparse algorithm of NMF, as well as NMF without sparsity in Section 5. This is done for both artificial and real data. Section 6 then presents the results obtained by applying the proposed setup on the aforementioned datasets. Finally, this work is concluded in Section 7.

2 Nonnegative Matrix Factorization

Nonnegative Matrix Factorization (NMF) is a method that decomposes a certain input data matrix into two factor matrices, where all elements of these two matrices need to be nonnegative. These nonnegativity constraints allow for an additive parts-based representation of the network, making it possible to not only identify latent structures in the data but also easily interpret these.

In this section, we firstly introduce the necessary notation and problem formulation of NMF and its classic loss function in Section 2.1. It is possible to represent the network better by incorporating different constraints and regularization terms to the classic NMF problem, but this will be further explained in Section 3. Finally, in Section 2.2 we review the different algorithms capable of solving the NMF loss function.

2.1 Notation and Problem Formulation

Throughout the paper the following notation is used. Matrices are recognised by bold capitals ($\mathbf{X}$) and when a bold lowercase letter is indexed by a certain lowercase letter it indicates a column of the matrix ($\mathbf{a}_i$). Next, column vectors are denoted by bold lowercase letters ($\mathbf{a}$) and a specific element of the vector by an indexed lowercase letter (a_i). The MoorePenrose inverse is denoted with the symbol †. Furthermore, sets are recognized by Fraktur letters ($\mathcal{A}$), a matrix subscripted with such letters ($\mathbf{X}_\mathcal{A}$) is the submatrix, where all its columns are the ones indexed by the elements of the set $\mathcal{A}$. A vector subscripted with such letters does as well ($\mathbf{X}_\mathcal{A}$). The usual symbols for set operations apply, such as $\setminus$ for deletion of a subset from a set, or $\cup$ for union of two sets. The ℓ_p-norm of any vector $\mathbf{v} \in \mathbb{R}^m$ is defined as $\|\mathbf{a}\|_p = \left(\sum_{j=1}^m |a_j|^p\right)^{\frac{1}{p}}$. Specifically, the ℓ_0-(pseudo)norm can be interpreted as the sum of all nonnegative elements $\sum_{j=1}^m \pi[a_j \neq 0]$ and is denoted as $\|\cdot\|_0$. The Frobenius norm is represented in a similar manner $\|\cdot\|_\mathrm{F}^2$

Matrix factorization is defined as the decomposition of an input matrix into a product of matrices. If we let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$ denote the input matrix with n data vectors of length m and a reduced rank r, then we can factorize matrix $\mathbf{X}$ into the product of matrices $\mathbf{W} = [w_{is}] \in \mathcal{R}^{n\times r}$ and $\mathbf{H} = [h_{js}] \in \mathcal{R}^{m\times r}$, such that

$$\mathbf{X} = \mathbf{W}\mathbf{H}^\top. \tag{1}$$

Its purpose can be viewed as finding a set of basis vectors, which approximates the columns of the data matrix $\mathbf{X}$ as closely as possible, where the basis vectors are then the columns of $\mathbf{W}$. Different metrics are capable of making a loss function from equation (1), such as Euclidean distance, KL-divergence, α/β-divergence, etc. However, since the Euclidean distance is the most widely used one, in this work we focus on the classic Euclidean distance as error measure (Fortunato and Hric, 2016).

The nonnegative matrix factorization (NMF) was first introduced by Paatero and Tapper (1994). Nonetheless, it was the work of Lee and Seung (1999), which established nonnegative matrix factorization as a method capable of a parts-based representation of the whole. It is different from other matrix factorization techniques, such as principal component analysis or singular value decomposition, mainly in its nonnegative constraints on all elements of the

output matrices $\mathbf{W}$ and $\mathbf{H}$. By including the nonnegativity constraints, the factorization of the matrix results in solely additive combinations of bases without any subtractions, which makes the result parts-based and thus more interpretable.

Nonnegative matrix factorization can be formalized as a minimization problem with a certain loss function $L_{\text{NMF}}(\mathbf{W}, \mathbf{H})$ and its nonnegative constraints. For a part-based representation of the whole, the nonnegativity constraints become $w_{is} \geq 0, h_{js} \geq 0$ for all elements of $\mathbf{W}$ and $\mathbf{H}$, respectively. For ease of notation we will throughout this work denote this as $\mathbf{W} \geq \mathbf{0}$ and $\mathbf{H} \geq \mathbf{0}$. The resulting optimization problem the becomes

$$\begin{aligned} \min_{\mathbf{W},\mathbf{H}} \quad & L_{\text{NMF}}(\mathbf{W}, \mathbf{H}) = \left\| \mathbf{X} - \mathbf{W}\mathbf{H}^\top \right\|_{\text{F}}^2 \\ \text{s.t.} \quad & \mathbf{W} \geq \mathbf{0}, \\ & \mathbf{H} \geq \mathbf{0}. \end{aligned} \tag{2}$$

One of the main advantages of using NMF is the ability to change the aforementioned loss-function to represent the data matrix as accurately as possible. The loss function can be altered by either changing the error loss term or by adding a regularization term, further explained in Section 3.

2.2 Algorithms

When both $\mathbf{W}$ and $\mathbf{H}$ are considered simultaneously, the optimization problem is difficult to solve due to its nonconvexity. In fact, it has been shown by Vavasis (2009) that the decision problem of NMF is NP-hard. This would result in meta-heuristic solvers, which do not guarantee an optimal result for solving the full NMF problem. Differently, by fixing one of the factor matrices, whilst optimizing the other, the subproblems become convex. The loss function then can be written as $L_{\text{NMF}}(\mathbf{W}|\mathbf{H})$ for fixing $\mathbf{H}$ and $L_{\text{NMF}}(\mathbf{H}|\mathbf{W})$ for fixing $\mathbf{W}$. So an alternating algorithm can be used to obtain at least locally optimal solutions for the NMF problem, as seen in Algorithm 1.

Algorithm 1: Alternating Updating

Initialization Set $k = 0$
Set $\mathbf{W}^{(0)}$ as a random initialization matrix with $w_{is} \geq 0$
Set $\mathbf{H}^{(0)}$ as a random initialization matrix with $h_{js} \geq 0$
while *convergence criterion not met* **do**
    Fix $\mathbf{H}^{(k)}$ find $\mathbf{W}^{(k+1)}$
    Fix $\mathbf{W}^{(k+1)}$ find $\mathbf{H}^{(k+1)}$
    Set $k = k + 1$
end
Return $\mathbf{W}^{(k+1)}$, $\mathbf{H}^{(k+1)}$

The main algorithm which is capable of optimally solving the subproblems for classic NMF is alternating nonnegative least squares. This algorithm makes use of nonnegative least squares (NNLS) by Lawson and Hanson (1995). One of the main disadvantages of it is its computational cost. Moreover, obtaining the optimal solutions to the subproblems of $\mathbf{W}$ and $\mathbf{H}$ does not equate to obtaining the optimal solution for the combination of both.

Therefore, research has focused on finding an approximate solution with less computation time. One such algorithm is multiplicative updating, which is a special case of projected gradient descent. Due to its simplicity, this algorithm has become very popular in literature (Fortunato, 2010). Both the alternating nonnegative least squares and the multiplicative updating algorithm are further explained in Section 2.2.1 and Section 2.2.2, respectively.

2.2.1 Alternating Nonnegative Least Squares

Alternating nonnegative least squares (ANLS) based on the active-set nonnegative least squares (NNLS) algorithm by Lawson and Hanson (1995) results in an optimal solution to the subproblem of $\mathbf{W}$ and $\mathbf{H}$. The active-set NNLS algorithm solves

$$\begin{aligned} \min_{x} \quad & L_{\text{NNLS}}(x) = \frac{1}{2} \left\| \mathbf{Cy} - \mathbf{b} \right\|_{\text{F}}^2 \\ \text{s.t.} \quad & y_i \geq \mathbf{0} \end{aligned} \tag{3}$$

where $\mathbf{C}$ is a $p \times v$ matrix, $\mathbf{b}$ a $v \times 1$ vector and $\mathbf{y}$ the unknown nonnegative $p \times 1$ vector. The active-set method by Lawson and Hanson (1995) solves this problem and is shown in Algorithm 2. As the name suggests, it makes use of an active set $\mathcal{Z}$ for elements that are still negative or zero, and a passive set $\mathcal{P}$ for positive elements.

Algorithm 2: Active-set NNLS

Initialization Set $\mathcal{Z} = \{1, 2, \ldots, v\}$
Set $\mathcal{P} = \varnothing$
Set $\mathbf{y} = \mathbf{0}$
Let $\mathbf{f} = \mathbf{C}^\top(\mathbf{b} - \mathbf{Cy})$
while $\mathcal{Z} = \varnothing$ and $\max_{i \in \mathcal{Z}}(\mathbf{f}) > 0$ **do**
    Let $j = \arg\max \mathbf{f}$
    Set $\mathcal{Z} \leftarrow \mathcal{Z} \setminus j$
    Set $\mathcal{P} \leftarrow \mathcal{P} \cup j$
    Let $\mathbf{z}_\mathcal{P} = \mathbf{G}_\mathcal{P}^\dagger \mathbf{b}$
    Let $\mathbf{z}_\mathcal{Z} = 0$
    while $j \in \mathcal{P}$ with $z_j < 0$ **do**
        Let $\alpha = \min_{\mathcal{K} \subset \mathcal{P}} \frac{\mathbf{y}_\mathcal{K}}{\mathbf{y}_\mathcal{K} - \mathbf{z}_\mathcal{K}}$
        Sey $\mathbf{y} = \mathbf{y} + \alpha(\mathbf{z} - \mathbf{y})$
        Set $\mathcal{Z} \leftarrow \{i | y_i = 0\}$
        Set $\mathcal{P} \leftarrow \{i | y_i > 0\}$
        Let $\mathbf{z}_\mathcal{P} = \mathbf{G}_\mathcal{P}^\dagger \mathbf{b}$
        Let $\mathbf{z}_\mathcal{Z} = 0$
    end
    Set $\mathbf{y} = \mathbf{z}$
end
Return $\mathbf{y}$

This algorithm is categorized by both an outer and inner while-loop (Luo and Duraiswami, 2011). In the outer loop, the largest positive element of the negative gradient

is moved from the active set to the passive set until the list is empty or the convergence criterion is met. For the inner loop, which runs through lines 10 to 17, the infeasible nonnegative solution moves towards a feasible one, where the nonnegative constraint is continuously satisfied. The new solution y_i is then moved from $\mathcal{P}$ to $\mathcal{Z}$ if $y_i = 0$. Lawson and Hanson (1995) show that this algorithm converges to the optimal solution and decreases monotonically for each iteration.

This algorithm can be implemented for NMF by fixing one of the two factor matrices, whilst the other is updated. Since the optimization problem then reduces to a NNLS problem in the form of $\|\mathbf{H}\mathbf{w}_i - \mathbf{x}_i\|_{\mathrm{F}}^2$ for fixed $\mathbf{H}$ and $\|\mathbf{W}\mathbf{h}_j - \mathbf{x}_j\|_{\mathrm{F}}^2$ for fixed $\mathbf{W}$. It must be noted that $\mathbf{h}_j$ is a column vector of the rows of $\mathbf{H}$. The vectors $\mathbf{w}_i$ and $\mathbf{h}_j$ can then be updated using the active set NNLS algorithm.

Nonetheless, this algorithm is computationally costly, as it needs to compute the Moore-Penrose inverse at every inner while loop. For NMF this algorithm also needs to be repeated for the complete matrix $\mathbf{W}$ and $\mathbf{H}$ (Luo and Duraiswami, 2011). Moreover, it can only be used for the minimization problem based on the Euclidean distance, which makes it unable to handle data with other distributions or alternative NMF loss functions.

2.2.2 Multiplicative Updating

The multiplicative updating algorithm was first introduced by Daube-Witherspoon and Muehllehner (1986) and made popular by Lee and Seung (2001). The updating rules are a special case of the gradient descent algorithm, where a predefined step size is set. It gives a quicker, but approximate solution to the NMF problem, since there is no proof that the algorithm converges to a stationary point (Lin, 2007). Nonetheless, it remains popular for its simplicity and ability to handle alternative loss functions of NMF, which will be discussed more thoroughly in Section 3.2.

Since this method is based on the projected gradient descent algorithm, per subproblem iteration it takes a step in the same direction as the negative gradient with a certain weight. It, therefore, iterates over the following condition

$$x^{(k+1)} = x^{(k)} - \alpha \frac{\partial f\left(x^{(k)}\right)}{\partial x^{(k)}}, \tag{4}$$

where $f(x)$ is the loss function and α is the size of the step towards the negative gradient. For the alternating NMF problems this becomes

$$w_{is}^{(k+1)} = w_{is}^{(k)} - \alpha_{w_{is}} \frac{\partial L_{\mathrm{NMF}}\left(w_{is}^{(k)}\right)}{\partial w_{is}^{(k)}} \tag{5}$$

for elements in $\mathbf{W}$ and

$$h_{js}^{(k+1)} = h_{js}^{(k)} - \alpha_{h_{js}} \frac{\partial L_{\mathrm{NMF}}\left(h_{js}^{(k)}\right)}{\partial h_{js}^{(k)}}, \tag{6}$$

for elements in $\mathbf{H}$.

To obtain the final multiplicative updating rules, both the gradient with respect to w_{is} and h_{js} is needed, as well as the step sizes for both. It is more straightforward to obtain these gradients in terms of matrices and then using an element-wise operator for the specific elements. The gradient with respect to $\mathbf{W}$ and $\mathbf{H}$ is obtained by rewriting the Frobenius

norm in terms of traces and expanding, such that

$$L_{\text{NMF}}(\mathbf{W},\mathbf{H}) = \|\mathbf{X}-\mathbf{W}\mathbf{H}^\top\|_{\text{F}}^2 = \operatorname{tr}\left((\mathbf{X}-\mathbf{W}\mathbf{H}^\top)(\mathbf{X}-\mathbf{W}\mathbf{H}^\top)^\top\right)$$
$$= \operatorname{tr}\mathbf{X}\mathbf{X}^\top + \operatorname{tr}\mathbf{W}\mathbf{H}^\top\mathbf{H}\mathbf{W}^\top - 2\operatorname{tr}\mathbf{X}\mathbf{H}\mathbf{W}^\top. \tag{7}$$

The partial derivative of the loss function with respect to $\mathbf{W}$ and $\mathbf{H}$, results in

$$\frac{\partial L_{\text{NMF}}(\mathbf{W},\mathbf{H})}{\partial \mathbf{W}} = -2\mathbf{X}\mathbf{H} + 2\mathbf{W}\mathbf{H}^\top\mathbf{H} \tag{8}$$

and

$$\frac{\partial L_{\text{NMF}}(\mathbf{W},\mathbf{H})}{\partial \mathbf{H}} = -2\mathbf{X}^\top\mathbf{W} + 2\mathbf{H}\mathbf{W}^\top\mathbf{W}, \tag{9}$$

respectively. The step sizes for $\mathbf{W}$ and $\mathbf{H}$ of the multiplicative updating algorithm by Lee and Seung (2001) are fixed and set equal to

$$\alpha_{\mathbf{W}} = \mathbf{W} \oslash \mathbf{W}\mathbf{H}^\top\mathbf{H}, \tag{10}$$

and

$$\alpha_{\mathbf{H}} = \mathbf{H} \oslash \mathbf{H}\mathbf{W}^\top\mathbf{W} \tag{11}$$

where $\oslash$ here is the element wise division, also known as the Hadamard division. By combining these conditions, we obtain the following multiplicative updating rules with element-wise operators

$$\mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} \circ \left(\mathbf{X}\mathbf{H} \oslash \mathbf{W}^{(k)}\mathbf{H}^\top\mathbf{H}\right) \tag{12}$$

and

$$\mathbf{H}^{(k+1)} = \mathbf{H}^{(k)} \circ \left(\mathbf{X}^\top\mathbf{W} \oslash \mathbf{H}^{(k)}\mathbf{W}^\top\mathbf{W}\right), \tag{13}$$

where $\circ$ is the element-wise multiplication, also known as the Hadamard product.

If we want to obtain the updating rules per element, we can rewrite the above expressions as elements instead of matrices. The updating rules then become

$$w_{is}^{(k+1)} = w_{is}^{(k)} \frac{\sum_j \left(x_{ij}h_{js} / \sum_s w_{is}^{(k)}h_{js}\right)}{\sum_j h_{js}}, \tag{14}$$

for w_{is} and

$$h_{js}^{(k+1)} = h_{js}^{(k)} \frac{\sum_i \left(x_{ij}^\top w_{is} / \sum_s h_{js}^{(k)}w_{is}\right)}{\sum_i w_{is}} \tag{15}$$

for h_{js}.

3 Nonnegative Matrix Factorization for Community Detection

In the previous section, we introduced the concept of nonnegative matrix factorization and stated that its loss function is easily adapted by weighting the error loss function or by adding a regularization term.

In this section, we explain how NMF can be used to solve the community detection problem. We therefore firstly discuss the notation and problem formulation of the community detection problem in Section 3.1. Afterwards, Section 3.2 discusses how nonnegative matrix factorization is capable of interpreting the community detection problem by creating a loss function, and how this loss function can be further changed for a better fit.

3.1 Notation and Problem Formulation

A complex network with n nodes and m links between the nodes can be formally represented by a graph. A graph is denoted as $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ consists of a non-empty set of vertices or nodes and $E = \{e_{ij} | v_i \in V \wedge v_j \in V\}$ denotes the set of all edges between two nodes, where an edge $e_{ij} = \{v_i, v_j\}$ is said to join the vertices v_i and v_j. These topological features are summarized in an adjacency matrix $\mathbf{A} = [a_{ij}]^{n \times n}$, with $a_{ij} = 1$ if nodes v_i and v_j are connected and 0 otherwise. A distinction can be made between directed and undirected networks. For directed networks the adjacency matrix $\mathbf{A}$ is asymmetric, whilst undirected networks have a symmetric adjacency matrix with all diagonal elements equal to 0. The neighbourhood of a node v_i in a certain graph $G_M = (V_M, E_M)$ is the subgraph of all nodes with edges adjacent to v_i, such that

$$N_M(v_i) = \{v \in V_M | : v_i, v \in E_M\}. \tag{16}$$

The degree of a node, denoted as $\delta(v_i)$, is equal to the number of its incident edges. It is formally determined by

$$\delta(v_i) = \sum_v \mathbb{1}_{\{\{u,v\} \in E\}} = |N_G(v_i)|, \tag{17}$$

where $\mathbb{1}_{\{\cdot\}}$ represents the indicator function.

The community detection problem is not clearly defined, due to the many different domain-specific applications. However, common convention underlies a difference between hard clustering and soft clustering.

Hard clustering is defined as the partitioning of nodes into non-overlapping communities, where a community is defined as a set of nodes with densely connected vertices within, and sparsely connected vertices between communities (Javed et al., 2018; Fortunato and Hric, 2016). More formally, let $\mathcal{C} = \{C_1, C_2, \ldots, C_K\}$ be a partition of V, where each cluster $C_k \, \forall k = 1, \ldots, K$ is non-empty. $\mathcal{C}$ is then the clustering of $\mathcal{G}$ with C_k the communities. The difference between hard clustering an soft clustering is that for hard clustering each node can only belong to a single cluster, such that $C_i \cap C_j = 0$, $\forall i, j : i \neq j$, $C_i, C_j \in \mathcal{C}$, whilst for soft clustering the nodes are allowed to overlap.

For soft clustering, the set of clusters is defined as $\mathcal{C} = \{C_1, C_2, \ldots, C_K\}$ and within it, nodes are capable of belonging to multiple communities (Lancichinetti et al., 2009). Moreover, a certain matrix $\mathbf{G} \in \mathcal{R}^{n \times K}$ is defined, where each element of the matrix g_{ik} denotes if node i belongs to a community k. A cover is then defined as $C_k = \{v_i \in V : g_{ik} \neq 0\}$.

3.2 Nonnegative Matrix Factorization for Community Detection

Wang et al. (2011) first used NMF as a community detection technique, after Ding et al. (2005) discovered that the NMF algorithm is closely related to k-means clustering and other (graph) partitioning methods. For community detection, NMF divides the network, which is represented by the adjacency matrix $\mathbf{A}$, into a given low-dimensional subspace through decomposition into a community feature matrix $\mathbf{W}$ and a community weighting matrix $\mathbf{H}$. The rank of these matrices is set equal to the number of desires communities K and each row represents an individual node. We can therefore state that $\mathbf{W} = [w_{is}]^{n \times K}$ and likewise $\mathbf{H} = [h_{js}]^{n \times K}$, where n is the previously defined total number of nodes. Using NMF as a community detection method, the columns of the feature matrix can be interpreted as bases of a low-dimensional subspace. Then each element from the weighing matrix is the contribution of this subspace to represent the given adjacency matrix. A node then belongs to the community where this weight is maximized. So the rows of the matrix $\mathbf{H}$ show the belonging of the nodes to communities, whilst the columns of $\mathbf{H}$ can be viewed as what nodes belong to what community.

More formally, each column of the adjacency matrix $\mathbf{a}_j$ are considered as the complete set of edges between the vertex v_j and the rest of the network $v_i \, \forall i \neq j$. Communities within a network can be represented by a new geometric subspace spanned by its vertices. Since we know that $\mathbf{W}$ can be viewed as such a basis matrix, where each column vector $\mathbf{w}_s$ denotes such a geometric subspace, a linear weighted combination of these basis approximates the network representation $\mathbf{a}_j$ as it is possible for vertices to belong to different communities. Each weight element h_{js} denotes the s^{th} element of row vector $\mathbf{h}_j^\top$ and indicates the contribution of basis vector $\mathbf{w}_s$ to the representation of $\mathbf{a}_j$, such that

$$\mathbf{a}_j \approx \sum_{s=1}^{K} \mathbf{w}_s h_{js}.$$

It is then indeed the case that for community detection, that

$$\mathbf{A} \approx \mathbf{W}\mathbf{H}^\top. \tag{18}$$

For hard clustering, index S of the largest element of row $\mathbf{h}_j^\top$ indicates the belonging of v_j to community k. Additionally, as all elements in $\mathbf{A}$ are strictly positive, nonnegative matrix factorization can be used for community detection, with the adjacency matrix $\mathbf{A}$ as input matrix.

3.2.1 Weighted Error Loss Function

As explained in Section 2, the classic NMF loss function uses the squared Frobenius norm and is in the form of $\|\mathbf{A} - \mathbf{W}\mathbf{H}^\top\|_{\text{F}}^2$. This norm inherently assumes Gaussian noise, which results in biased estimates in the presence of outliers or non-Gaussian noise (Huber, 1964). As noise is characteristic of real-world networks, it is important to have an objective function that is robust to outliers. The norm of the loss function is often changed to allow for a more robust

objective function. In this section, we briefly mention the most common types of weighting schemes for robustifying the loss function. To account for different types of weighting on this error loss function, it can be rewritten as a summation of residuals, such that

$$L_{\text{WNMF}} = \sum_{j=1}^{n}\sum_{i=1}^{n} \omega_{ij} \left(a_{ij} - \mathbf{w}_i^\top \mathbf{h}_j\right)^2 = \sum_{j=1}^{n}\sum_{i=1}^{n} \omega_{ij}\epsilon_{ij}^2, \tag{19}$$

where ω_{ij} can be any weight given to the error loss. Note that classic NMF is a special case of this weighted loss function, i.e. when all weights are equal to 1.

ℓ_1-Robust NMF To improve robustness of outliers and noise of the loss function, De La Torre and Black (2003) used the ℓ_1-norm for the NMF loss function. This results in the optimization problem of

$$\begin{aligned} \min_{\mathbf{W},\mathbf{H}} \quad & L_{\ell_1-\text{RNMF}} = \left\|\mathbf{A} - \mathbf{W}\mathbf{H}^\top\right\|_1 = \sum_{j=1}^{n}\sum_{i=1}^{n} |a_{ij} - \mathbf{w}_i^\top \mathbf{h}_j| \\ \text{s.t.} \quad & \mathbf{W} \geq \mathbf{0},\ \mathbf{H} \geq \mathbf{0}. \end{aligned} \tag{20}$$

As the sum of the absolute values of the loss function is computed, the resulting NMF minimization problem is a non-smooth optimization problem. Many algorithms have intended to efficiently solve the robust ℓ_1-norm loss function, such as Eriksson and van den Hengel (2012), Zheng et al. (2012) and Shen et al. (2014). However, algorithms that solve this optimization problem either have no theoretical convergence guaranteea, or lack scalability (Lin et al., 2017).

$\ell_{2,1}$-Robust NMF The less-known $\ell_{2,1}$-norm, first introduced by Kong et al. (2011), can also be used to make the loss function more robust. The $\ell_{2,1}$-norm for a certain $n \times m$ matrix is defined as $\|\mathbf{X}\|_{2,1} = \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{m} x_{ij}^2} = \sum_{i=1}^{n} \|x_i\|_2$. A more intuitive way of viewing the $\ell_{2,1}$-norm is by first computing the ℓ_2-norm of the rows w_i of the data matrix. Afterwards, the results are appended in a vector $\mathbf{v}$, such that $\mathbf{v} = [\|x_1\|_2, \|x_2\|_2, \ldots, \|x_n\|_2]$. The magnitude of the components in this vector determine the importance of each dimension (Yang et al., 2013). As such, by computing the ℓ_1-norm of this resulting vector $\mathbf{v}$, only the most important components are considered. This change still allows for convexity of the problem, whilst not squaring the errors. The $\ell_{2,1}$ objective function is written as

$$\begin{aligned} \min_{\mathbf{W},\mathbf{H}} \quad & L_{\ell_{2,1}-\text{RNMF}} = \left\|\mathbf{A} - \mathbf{W}\mathbf{H}^\top\right\|_{2,1} = \sum_{i=1}^{n} \left(\sum_{j=1}^{m} \left(a_{ij} - \mathbf{w}_i^\top \mathbf{h}_j\right)^2\right)^{1/2} \\ \text{s.t.} \quad & \mathbf{W} \geq \mathbf{0},\ \mathbf{H} \geq \mathbf{0}. \end{aligned} \tag{21}$$

Nonetheless, this norm is less intuitive than either the classic squared Frobenius norm or ℓ_1-norm. It is also less capable of handling noise when compared to the ℓ_1-norm. However, Bai (1995) proofed that the ℓ_1-norm becomes inefficient when data are normally distributed.

Huber-Robust NMF A weighting scheme capable of resolving these issues is the Huber function, which was first introduced by Huber (1964). This results in the following optimization problem

$$\begin{aligned} \min_{\mathbf{W},\mathbf{H}} \quad & L_{\text{Huber}-\text{RNMF}} = \begin{cases} \sum_{i=1}^{n}\sum_{j=1}^{m}\left(a_{ij}-\mathbf{w}_i^\top\mathbf{h}_j\right)^2, & \text{if } |a_{ij}-\mathbf{w}_i^\top\mathbf{h}_j| \le c \\ \sum_{i=1}^{n}\sum_{j=1}^{m} 2c\left(|a_{ij}-\mathbf{w}_i^\top\mathbf{h}_j|-c\right), & \text{if } |a_{ij}-\mathbf{w}_i^\top\mathbf{h}_j| \ge c \end{cases} \\ \text{s.t.} \quad & \mathbf{W}\ge\mathbf{0},\ \mathbf{H}\ge\mathbf{0}. \end{aligned} \tag{22}$$

The constant c is a trade-off parameter which ensures a larger amount of robustness for smaller values of the constant. For NMF, the median of the absolute value of all residuals ϵ_{ij} is often chosen Du et al. (2012). The Huber function becomes the classic Euclidean distance or ℓ_2-norm for small residuals, whilst for residuals larger than the threshold c, the robuster ℓ_1-norm is obtained. Due to this method, it is possible to combine the advantages of both the ℓ_2-norm and the ℓ_1-norm.

We can summarize the alternative loss functions from residuals $a_{ij}-\mathbf{w}_i^\top\mathbf{h}_j$ into Figure 1. We see that the loss from using the $\ell_{2,1}$-norm in a one-dimensional setting is the same as the ℓ_1-norm. It must be stated that this only holds for this one-dimensional example. For the Huber loss function we notice that it indeed follows the ℓ_2-norm until $c = 1.5$ and then it follows the ℓ_1-norm

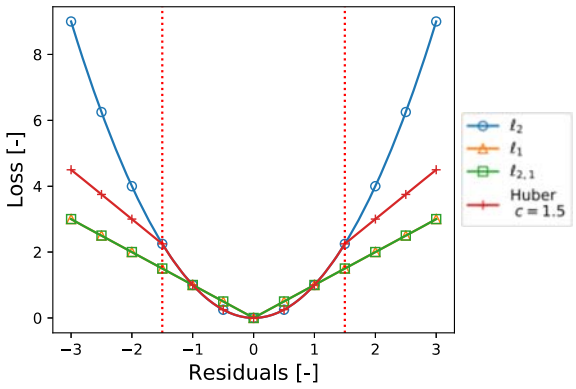


Figure 1: One-dimensional example of the loss function for different robust weights, where the dotted line represents the constant c of the huber function.

3.2.2 Graph Regularization

Separate from the error loss function, different regularization constraints can be added to the overall loss function. One such regularization term is graph regularization, which has gained popularity, as it significantly improves the performance accuracy of the community detection task (Cai et al., 2010; Wu et al., 2018).

By adding a graph regularization term in the loss function, as first explained by Cai et al. (2010), update pushes the solution to resemble the local structure of the network. It based on the assumption that geometrically similar nodes in a network tend to have similar

characteristics. Since NMF tries to find basis vectors able to represent the data as close as possible, graph regularization not only conserves geometric similarity in Euclidean space but also a non-Euclidean one. This latter makes it possible for edges to be curved as well. Since the network representation is bound to fewer assumptions, less information loss of the network occurs. Cai et al. (2010) make use of nearest neighbour graphs to model and measure this local geometric similarity, such that nodes that are both Euclidean and non-Euclidean close to each other in the network, remain close in the latent subspace of the network. This idea is represented by the following loss function:

$$\begin{aligned} \min_{\mathbf{W},\mathbf{H}} \quad & L_{\text{GNMF}} = \left\|\mathbf{A} - \mathbf{W}\mathbf{H}^\top\right\|_{\text{F}}^2 + \frac{\lambda}{2}\sum_i \sum_j q_{ij} \left\|\mathbf{h}_i - \mathbf{h}_j\right\|^2 \\ \text{s.t.} \quad & \mathbf{W} \geq \mathbf{0},\ \mathbf{H} \geq \mathbf{0} \end{aligned} \tag{23}$$

where q_{ij} represents the similarity between a certain vertex pair v_i and v_j. It is thus possible to incorporate prior information of the pairwise relationship between vertices. This amount is controlled by the regularization parameter λ. As an example, when $q_{ij} = 1$, it is known beforehand that v_i and v_j belong in the same community. Then by minimizing (23), the regularization term pushes the two vertices into the same subspace representation.

For ease of notation, the graph regularization term is often described by the trace. By denoting $\mathbf{Q}$ as a matrix with elements q_{ij}, such that $\mathbf{Q} = [q_{ij}] \in \mathbb{R}^{n\times n}$, we can construct a diagonal matrix $\mathbf{D_Q}$ whose elements d_{ii} are determined by the row summation of $\mathbf{Q}$, namely $d_{ii} = \sum_{j=1}^{n} q_{ij}$. The Laplacian of this weighting matrix $\mathbf{Q}$ is defined as $\mathbf{L_Q} = \mathbf{D_Q} - \mathbf{Q}$, such that we can rewrite the graph regularization term as $\lambda \operatorname{tr}(\mathbf{H}^\top \mathbf{L_Q} \mathbf{H})$. The optimization problem then becomes

$$\begin{aligned} \min_{\mathbf{W},\mathbf{H}} \quad & L_{\text{GNMF}} = \left\|\mathbf{A} - \mathbf{W}\mathbf{H}^\top\right\|_{\text{F}}^2 + \lambda \operatorname{tr}(\mathbf{H}^\top \mathbf{L_Q} \mathbf{H}) \\ \text{s.t.} \quad & \mathbf{W} \geq \mathbf{0},\ \mathbf{H} \geq \mathbf{0}. \end{aligned} \tag{24}$$

If we take as similarity measure a simple *Binary Weighting* scheme, where $q_{ij} = 1$ when vertex i is connected with vertex j and zero otherwise, the weighting matrix $\mathbf{Q}$ is then the adjacency matrix $\mathbf{A}$. This results in the actual graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$. When the *Dot-Product Weighting* is used, the weights become $q_{ij} = a_i^\top a_j$ if vertex v_i and v_j are connected. This weighting is the same as the cosine similarity of the two vertices when normalizing a_i and a_j.

Nonetheless, graph regularization is limited to the assumption that an edge is only capable of connecting two vertices. Due to this, it only incorporates pairwise relations between nodes, which tends to oversimplify relations in complex networks. Instead of using graph learning, a novel approach is to use hypergraph learning, introduced by Zhou et al. (2007). This method discards this pairwise assumption of graphs and makes use of hypergraphs, which have edges that can have a connection between two *or more* vertices. The only thing that changes for the regularization term is how the Laplacian is obtained, as hypergraphs are used instead of graphs. Nonetheless, the dimension of this hypergraph Laplacian is still equal to $n \times n$ and therefore does not change algorithms used for solving the optimization problem.

3.2.3 Sparsity Regularization

Sparsity regularization can be added to the NMF loss function to directly control the sparseness of the problem. If it is added to the full matrix of $\mathbf{H}$ or $\mathbf{W}$, then the term helps to avoid a general over-fitting on the network data. Nonetheless, for community detection, where the rows and columns of these matrices have a certain meaning, sparsity is imposed on the rows and columns instead of the full matrix.

Without imposing sparsity on the NMF problem, the nonnegative constraints still allow for some sparsity. This is mainly through the algorithms that solve the NMF loss function. For example, the multiplicative updating algorithm will continue having a value equal to zero for all iterations once an update finds a result equal to zero. Nonetheless, as it is not made explicit in the loss function, it does not have a trade-off parameter able to control the amount (Peharz and Pernkopf, 2012; Laurberg, 2008).

For community detection Kim and Park (2008) state that it is the rows of the matrix $\mathbf{H}$ that need to be made sparse. This results in each column of $\mathbf{A}$ being represented by only a limited number of communities. As such, a vertex v_j is only capable of belonging to a limited number of communities. For classic NMF, the reduced rank r of $\mathbf{A}$, is set as the number of communities K, where $K << n$. By imposing the sparsity constraint, we do not need to set $r = K$, instead, we can choose r as any real positive integer larger than K. In theory, it is also possible to impose such a sparsity constraint on columns of $\mathbf{H}$, which restricts the number of vertices capable of belonging to a certain community. As this specific constraint does not often occur in the community detection task, it is not further discussed.

Peharz and Pernkopf (2012) argue that in some cases a sparsity restriction on columns of $\mathbf{W}$ is desired, as it results in the formation of subspaces with limited interconnectedness. Due to this, a more disjoint view of the network is encouraged, which enhances the parts-based representation. A constraint on the rows of $\mathbf{W}$ is not discussed, as this constraint has no inherent meaning.

From the above, we can conclude that for the community detection task, it is desired to have either a sparsity constraint in rows of $\mathbf{H}$ or columns of $\mathbf{W}$. By denoting F as the number of nonzero elements, the following optimization problems arise

$$\begin{aligned} \min_{\mathbf{W},\mathbf{H}} \quad & L_{\text{H-SNMF}} = \left\|\mathbf{A} - \mathbf{W}\mathbf{H}^\top\right\|_{\text{F}}^2 \\ \text{s.t.} \quad & \mathbf{W} \geq \mathbf{0}, \\ & \mathbf{H} \geq \mathbf{0}, \\ & \|\mathbf{h}_j\|_0 \leq F \ \ \forall j = 1, \dots n \end{aligned} \tag{25}$$

for rows of $\mathbf{H}$ and

$$\begin{aligned} \min_{\mathbf{W},\mathbf{H}} \quad & L_{\text{W-SNMF}} = \left\|\mathbf{A} - \mathbf{W}\mathbf{H}^\top\right\|_{\text{F}}^2 \\ \text{s.t.} \quad & \mathbf{W} \geq \mathbf{0}, \\ & \mathbf{H} \geq \mathbf{0}, \\ & \|\mathbf{w}_i\|_0 \leq F \ \ \forall i = 1, \dots n \end{aligned} \tag{26}$$

for columns of $\mathbf{W}$. Since alternating algorithms are used for updating NMF, these problems become similar to solve (Peharz and Pernkopf, 2012).

ℓ_1-Sparse NMF Since the ℓ_1-norm is a convex relaxation of the ℓ_0-norm, Kim and Park (2008) use the ℓ_1-norm on rows of matrix $\mathbf{H}$ to enforce sparsity solely as in (25). They use the findings of Donoho and Elad (2003), which showed that it is indeed possible to obtain a sparser subset by using the ℓ_1-norm on the complete matrix $\mathbf{H}$ and $\mathbf{W}$. Kim and Park (2008) therefore introduce the following optimization problem

$$\begin{aligned} \min_{\mathbf{W},\mathbf{H}} \quad & L_{\ell_1\text{-SNMF}} = \left\|\mathbf{A} - \mathbf{W}\mathbf{H}^\top\right\|_{\mathrm{F}}^2 + \lambda \sum_{j=1}^{n} \|\mathbf{h}_j\|_1 \\ \text{s.t.} \quad & \mathbf{W} \geq \mathbf{0}, \\ & \mathbf{H} \geq \mathbf{0}, \end{aligned} \tag{27}$$

where $\mathbf{h}_j$ is defined as the column vector of the rows of $\mathbf{H}$. This optimization problem is solved by Morup et al. (2008), which use the nonnegative variant of least angle regression and selection (LARS), introduced by Efron et al. (2004), to impose sparsity. The algorithm returns a full set of solutions to (27) for different regularization values. The value that best approximates the given number of nonzero elements is then chosen.

Nonetheless, LARS is more sensitive to noise than other methods, as it iteratively refits the residuals. Additionally, as it is unclear what exact relation the hyperparameter has to the number of non-zero elements, the result is less interpretable and the algorithm is more difficult to tune (Kim and Park, 2008).

For all ℓ_1-norm problems sparsity regularizers it holds that a more intuitive method of enforcing sparsity in community detection is the use of the ℓ_0-norm, since the problem is constraining the number of nonzero entries in rows of $\mathbf{H}$. Many argue that the ℓ_0-norm is non-convex and solving the problem is NP-hard. However, the nonnegative matrix factorization problem is already inherently NP-hard by itself, thus constraining the ℓ_0-norm of the factor matrices to a certain allowable number will not change the nature of the NMF problem.

NMF with ℓ_0 constraint Little research has been done regarding the use of the ℓ_0-(pseudo)norm as a sparsity constraint for NMF. The method proposed by Aharon et al. (2005) can be defined as NMF with ℓ_0-norm sparseness constraint on the columns of the factor matrix $\mathbf{W}$, which represents the subspace basis. However, the nonnegativity restriction is not considered whilst updating the factor matrices but only afterwards by setting the negative value to zero, resulting in non-optimality. Moreover, the manner of enforcing sparsity in the algorithm results in an overfitted solution (Peharz and Pernkopf, 2012).

Peharz and Pernkopf (2012) introduced the only benchmark method capable of enforcing ℓ_0-sparsity for NMF, the classic nonnegative matrix factorization with ℓ_0-constraints (ℓ_0-cNMF). This algorithm uses a combination of NNLS orthogonal matching pursuit (OMP), to obtain a nonnegative constrained OMP algorithm. It starts by randomly initializing $\mathbf{W}$. Afterwards, the algorithm is characterized by two steps, the nonnegative sparse coding step and the update of $\mathbf{W}$. For the first step, it sparsely encodes the data from $\mathbf{A}$, using OMP. As such, it performs an iterative step-wise selection of basis vectors with highest correlation between $\mathbf{A}$ and $\mathbf{W}$. The network is then represented by a projection onto the space spanned by these basis vectors, which gives in a sparser sub-space representation of the network. It, therefore, results in an ℓ_0-constrained nonnegative update for $\mathbf{H}$. For the second step, which is the update of $\mathbf{W}$, NNLS is used.

However, this method has a certain limit to the number of non-zero elements it can impose, which often results in less accurate results for larger datasets (Cai and Wang, 2011). Additionally, since in the sparse encoding step a sparser representation of $\mathbf{A}$ is obtained, this algorithm will perform worse for networks with high amount of sparsity in $\mathbf{A}$ (Needell and Vershynin, 2009). Moreover, it is difficult to obtain results for alternative NMF loss functions, such as graph regularized NMF. Since graph regularization allows for a better representation of the graph structure, it is important that this information is considered when enforcing ℓ_0-sparsity.

As all methods capable of enforcing sparsity for the community detection task, are either limited in their flexibility of the loss function or solely approximate the sparsity with the ℓ_1-norm, we notice that there is a need for a method which is capable of enforcing ℓ_0-sparsity on the loss function, whilst maintaining the flexibility of NMF. Therefore in the following section, we present a novel algorithm, which is capable of enforcing this ℓ_0-sparsity for community detection, whilst allowing for a different norm to be chosen in the error loss function or the addition of graph regularization.

4 Methodology

In the previous section, we concluded that there is a need for a method capable of enforcing ℓ_0-sparsity on the solution, whilst keeping the capability of having alternative loss functions of NMF. The purpose of this section is to derive a method capable of this.

Firstly, we explain the iterative majorization algorithm in Section 4.1, which is used for the solution. Afterwards in Section 4.2, ℓ_0-sparsity is enforced on the classic NMF loss function. Additionally, the novel algorithm is also capable of enforcing the ℓ_0-constraint on all alternative loss functions previously discussed in Section 3.2. This includes any change in the error loss function or the addition of graph regularization. However, we do not investigate the performance of these alternative versions, since we want to focus on its competitiveness in comparison to similar existing algorithms. The only algorithm which is capable of enforcing an ℓ_0-sparsity constraint is ℓ_0-cNMF, which can not handle these alternative loss functions. Therefore, we do not test its performance and only show how the updating rules are obtained for a weighted error loss function in Appendix A.1 and a weighted error loss function with graph regularization in Appendix A.2. How all versions of the novel algorithm can be implemented in an actual algorithm is clarified in Section 4.3. We explain how the performance of solely the classic novel algorithm is evaluated in section 4.5.

4.1 Iterative Majorization

Iterative Majorization (IM), often called Majorization Minimization, was first introduced by De Leeuw (1977) as a method for efficiently solving non-convex, non-smooth optimization problems. Adapted from Groenen et al. (2003), we can formally denote the iterative majorization algorithm. Given that the optimization function L is continuous, such that $L : \mathcal{X} \to \mathbb{R}$ with $\mathcal{X} \subseteq \mathbb{R}^d$, a majorization function $g(\mathbf{X}, \mathbf{X}^{(k)})$ can be constructed, where the superscript (k) denotes the current iteration. This function can be viewed as a surrogate function of the more complex problem $L(\mathbf{X})$, which must touch the optimization function at current iteration $\mathbf{X}^{(k)}$. Additionally, the majorization function should always be strictly larger or equal to the loss function. By design, the function for $g(\mathbf{X}, \mathbf{X}^{(k)})$ should be chosen such that the minimum is relatively simpler to compute when compared to the optimization function. Thus, a majorization function needs to satisfy the following conditions:

$$L(\mathbf{X}^{(k)}) = g(\mathbf{X}^{(k)}, \mathbf{X}^{(k)})$$
$$L(\mathbf{X}) \leq g(\mathbf{X}, \mathbf{X}^{(k)}) \; \forall \mathbf{X} \in \mathcal{X}$$

To find the minimum of $L(\mathbf{X})$, we construct the majorization function with starting point at $\mathbf{X}_r$ and thus setting the support point $\mathbf{X}^{(k)}$ equal to this random start. The subsequent iteration step can be found by minimizing the following $\mathbf{X}^+ = \arg\min_{\mathbf{X}} g(\mathbf{X}, \mathbf{X}^{(k)})$. Algorithm 3 depicts the pseudocode for iterative majorization and Figure 2 gives a two-dimensional example on the workings of this algorithm.

Using the properties of the majorization function, the following sandwich inequality holds $L(\mathbf{X}^+) \leq g(\mathbf{X}^+, \mathbf{X}^{(k)}) \leq g(\mathbf{X}^{(k)}, \mathbf{X}^{(k)}) = L(\mathbf{X}^{(k)})$. As the iterative majorization algorithm chooses supporting point $\mathbf{X}^{(k)}$ as the previous obtained minimum $\mathbf{X}^+$, the algorithm is guaranteed to be monotonically decreasing and thus guaranteed to converge to a local minimum if the majorization function is differentiable.

Algorithm 3: Iterative Majorization

Initialization Set $k = 0$
Set $\mathbf{X}^{(k)} = \mathbf{X}_r$ with $\mathbf{X}_r$ random start
while $L(\mathbf{X}^{(k)}) - L(\mathbf{X}^{(k+1)}) < \epsilon L(\mathbf{X}^{(k+1)})$ **do**
    Find $\mathbf{X}^{(k+1)} = \arg\min_{\mathbf{X}} g(\mathbf{X}, \mathbf{X}^{(k)})$
    Set $k = k + 1$
    Determine $L\left(\mathbf{X}^{(k+1)}\right)$
end
Return $\mathbf{X}^{+} = \mathbf{X}^{(k+1)}$

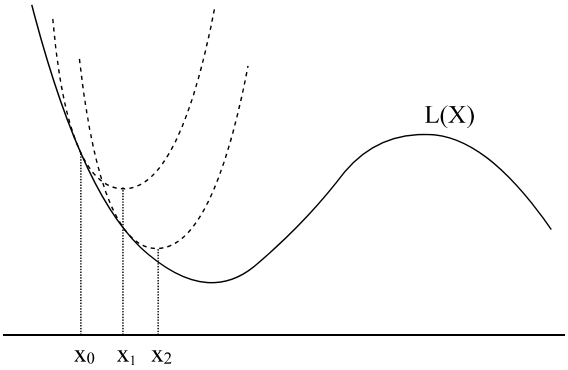


Figure 2: Two-dimensional example of the Iterative Majorization algorithm with a quadratic majorization function, modified from Sun et al. (2016).

4.2 ℓ_0-Sparse Nonnegative Matrix Factorization (ℓ_0-SNMF)

The necessary majorization functions to enforce sparsity on the nonnegative matrix factorization loss functions are derived here. In the previous section we stated that for community detection, it is desirable to either enforce sparsity on rows of $\mathbf{H}$, or columns of $\mathbf{W}$. Moreover, we want the novel algorithm to be capable of handling different loss functions. This means that the novel algorithm should be capable of handling the classic NMF loss function, but also different weighted error loss functions and graph regularization terms.

We have stated before that these problems become similar to solve, as alternating algorithms are used for updating. Nonetheless, when including a graph regularization term, which depends on $\mathbf{H}$, it is more difficult to obtain an updating rule for the novel algorithm. We therefore choose to further derive and study solely the enforcement of sparsity constraints on rows of $\mathbf{H}$.

In this section, we derive the updating rule of the novel algorithm for the classic error loss function. It is also possible to derive such an expression for a weighted error loss function and a graph regularized weighted error loss function. To our knowledge, no algorithm exists which can handle both an ℓ_0-sparsity constraint and different loss functions. Since this makes it difficult to compare our algorithm to any other benchmark, we will not test the performance of these algorithms and further research is needed to determine the performance and limitations of these algorithms. We solely give the derivations of these, which can be

found in Appendix A.1 and A.2, respectively.

First, we start with the classic error loss function, which does not include robust weights for the loss function, nor (mixed hyper)graph regularization. The optimization problem which we intend to solve is shown in (28), where $\|\mathbf{h}_j\|_0$ is the ℓ_0-(pseudo)norm taken on rows of $\mathbf{H}$.

$$\begin{aligned} \min \quad & L_{\ell_0\text{-SNMF}}\left(\mathbf{W},\mathbf{H}|\mathbf{A}\right) = \left\|\mathbf{A}-\mathbf{W}\mathbf{H}^\top\right\|_{\mathrm{F}}^2 \\ \text{s.t.} \quad & \mathbf{W} \geq \mathbf{0}, \\ & \mathbf{H} \geq \mathbf{0}, \\ & \|\mathbf{h}_j\|_0 \leq F, \end{aligned} \tag{28}$$

This work aims to enforce sparsity in the network through an iterative majorization approach to obtain an ℓ_0-Sparse Nonnegative Matrix Factorization (ℓ_0-SNMF) algorithm. More specifically, we want each row in $\mathbf{H}$ to have maximally F nonzero entries, which results in a data entry only belonging to F communities simultaneously. In this section, we show that it is possible to write optimization problem (28), as a sorting problem.

Since any solver of NMF uses alternating updating, we know that either $\mathbf{W}$ or $\mathbf{H}$ is solved, both with its corresponding constraints. If we add the ℓ_0-constraint on rows of $\mathbf{H}$, it will not have any effect on the update of $\mathbf{W}$, meaning that it is solved using any classic NMF solver. Nonetheless, we refer to Section 2.2 for an explanation on different standard updating rules for $\mathbf{W}$. In this section, we will therefore only focus on determining a method for updating $\mathbf{H}$ with the ℓ_0-constraint.

To enforce the constraint on $\mathbf{H}$, we need to write optimization problem (28) for unknown $\mathbf{H}$ and known $\mathbf{A}$ and $\mathbf{W}$. To do this, we first write the Frobenius norm of the classical NMF loss function as a trace, such that

$$\begin{aligned} L_{\mathrm{NMF}}(\mathbf{H}|\mathbf{A},\mathbf{W}) &= \left\|\mathbf{A}-\mathbf{W}\mathbf{H}^\top\right\|_{\mathrm{F}}^2 = \operatorname{tr}\left(\mathbf{A}-\mathbf{W}\mathbf{H}^\top\right)\left(\mathbf{A}-\mathbf{W}\mathbf{H}^\top\right)^\top \\ &= \operatorname{tr}\mathbf{A}\mathbf{A}^\top + \operatorname{tr}\mathbf{W}\mathbf{H}^\top\mathbf{H}\mathbf{W}^\top - 2\operatorname{tr}\mathbf{A}\mathbf{H}\mathbf{W}^\top. \end{aligned} \tag{29}$$

Using the cyclic property of traces, we can rewrite $\operatorname{tr}\mathbf{W}\mathbf{H}^\top\mathbf{H}\mathbf{W}^\top$ as $\operatorname{tr}\mathbf{H}\mathbf{W}^\top\mathbf{W}\mathbf{H}^\top$. We can majorize this rewritten term by observing that if $\gamma_{\max}$ denotes the largest eigenvalue of $\mathbf{W}^\top\mathbf{W}$, then by definition $\mathbf{W}^\top\mathbf{W}-\gamma_{\max}\mathbf{I}$ is negative semidefinite. Consequently, as first used in the context of majorization by Bijleveld and De Leeuw (1991), we can use this for majorization since

$$\operatorname{tr}\left(\mathbf{H}-\mathbf{H}^{(k)}\right)\left(\mathbf{W}^\top\mathbf{W}-\gamma_{\max}\mathbf{I}\right)\left(\mathbf{H}-\mathbf{H}^{(k)}\right)^\top \leq 0. \tag{30}$$

It is possible to choose any value for $\mathbf{H}^{(k)}$, we set $\mathbf{H}^{(k)}$ equal to the previous iteration result for $\mathbf{H}$ and thus known. By further expanding this expression, we obtain

$$\begin{aligned} &\operatorname{tr}\left(\mathbf{H}-\mathbf{H}^{(k)}\right)\left(\mathbf{W}^\top\mathbf{W}-\gamma_{\max}\mathbf{I}\right)\left(\mathbf{H}-\mathbf{H}^{(k)}\right)^\top \leq 0 \\ &\qquad \operatorname{tr}\left(\mathbf{H}-\mathbf{H}^{(k)}\right)\mathbf{W}^\top\mathbf{W}\left(\mathbf{H}-\mathbf{H}^{(k)}\right)^\top \leq \gamma_{\max}\operatorname{tr}\left(\mathbf{H}-\mathbf{H}^{(k)}\right)\left(\mathbf{H}-\mathbf{H}^{(k)}\right)^\top \end{aligned}$$

with final result after expanding and rearranging

$$\begin{aligned} \operatorname{tr}\mathbf{H}\mathbf{W}^\top\mathbf{W}\mathbf{H}^\top \leq{}& \gamma_{\max}\operatorname{tr}\mathbf{H}\mathbf{H}^\top - 2\gamma_{\max}\operatorname{tr}\mathbf{H}\mathbf{H}^{(k)\top} + 2\operatorname{tr}\mathbf{H}\mathbf{W}^\top\mathbf{W}\mathbf{H}^{(k)\top} \\ &+ \gamma_{\max}\operatorname{tr}\mathbf{H}^{(k)}\mathbf{H}^{(k)\top} - \operatorname{tr}\mathbf{H}^{(k)}\mathbf{W}^\top\mathbf{W}\mathbf{H}^{(k)\top}. \end{aligned} \tag{31}$$

By plugging this inequality constraint into the expanded loss function of (29) we get the following majorization function

$$\begin{aligned} L_{\text{NMF}}(\mathbf{H}|\mathbf{A},\mathbf{W}) &= \operatorname{tr}\mathbf{H}\mathbf{W}^\top\mathbf{W}\mathbf{H}^\top - 2\operatorname{tr}\mathbf{A}\mathbf{H}\mathbf{W}^\top + \operatorname{tr}\mathbf{A}\mathbf{A}^\top \\ &\leq \gamma_{\max}\operatorname{tr}\mathbf{H}\mathbf{H}^\top - 2\gamma_{\max}\operatorname{tr}\mathbf{H}\mathbf{H}^{(k)\top} + 2\operatorname{tr}\mathbf{H}\mathbf{W}^\top\mathbf{W}\mathbf{H}^{(k)\top} \\ &\quad + \gamma_{\max}\operatorname{tr}\mathbf{H}^{(k)}\mathbf{H}^{(k)\top} - \operatorname{tr}\mathbf{H}^{(k)}\mathbf{W}^\top\mathbf{W}\mathbf{H}^{(k)\top} - 2\operatorname{tr}\mathbf{A}\mathbf{H}\mathbf{W}^\top + \operatorname{tr}\mathbf{A}\mathbf{A}^\top \\ &= g_{\ell_0\text{NMF}}\left(\mathbf{H}|\mathbf{A},\mathbf{W},\mathbf{H}^{(k)}\right) \end{aligned} \tag{32}$$

Since $\mathbf{H}$ is the only unknown in $g_{\ell_0\text{NMF}}(\mathbf{H}|\cdot)$, where the dot represent all known variables, we can combine like terms for $\mathbf{H}$ and join all known terms into constant c_1 to obtain

$$\begin{aligned} g_{\ell_0\text{NMF}}(\mathbf{H}|\cdot) &= \gamma_{\max}\operatorname{tr}\mathbf{H}\mathbf{H}^\top - 2\gamma_{\max}\operatorname{tr}\mathbf{H}\left(\mathbf{H}^{(k)} - \frac{1}{\gamma_{\max}}\mathbf{H}^{(k)}\mathbf{W}^\top\mathbf{W} + \frac{1}{\gamma_{\max}}\mathbf{A}^\top\mathbf{W}\right)^\top \\ &\quad + \operatorname{tr}\mathbf{A}\mathbf{A}^\top + \gamma_{\max}\operatorname{tr}\mathbf{H}^{(k)}\mathbf{H}^{(k)\top} - \operatorname{tr}\mathbf{H}^{(k)}\mathbf{W}^\top\mathbf{W}\mathbf{H}^{(k)\top} \\ &= \gamma_{\max}\operatorname{tr}\mathbf{H}\mathbf{H}^\top - 2\gamma_{\max}\operatorname{tr}\mathbf{H}\mathbf{U}^\top + c_1, \end{aligned} \tag{33}$$

where

$$\mathbf{U} = \mathbf{H}^{(k)} - \frac{1}{\gamma_{\max}}\mathbf{H}^{(k)}\mathbf{W}^\top\mathbf{W} + \frac{1}{\gamma_{\max}}\mathbf{A}^\top\mathbf{W} \tag{34}$$

and

$$c_1 = \operatorname{tr}\mathbf{A}\mathbf{A}^\top + \gamma_{\max}\operatorname{tr}\mathbf{H}^{(k)}\mathbf{H}^{(k)\top} - \operatorname{tr}\mathbf{H}^{(k)}\mathbf{W}^\top\mathbf{W}\mathbf{H}^{(k)\top}. \tag{35}$$

The final expression in (33) resembles $\|\mathbf{H}-\mathbf{U}\|_{\text{F}}^2$ when expanded and rewritten in terms of traces. Nonetheless, the term $\gamma_{\max}\operatorname{tr}\mathbf{U}\mathbf{U}^\top$ is then missing, such that we need to subtract this term to write it as a Frobenius norm and obtain the final loss function of our ℓ_0-SNMF algorithm

$$L_{\ell_0\text{-SNMF}}(\mathbf{H}|\mathbf{U}) = \gamma_{\max}\|\mathbf{H}-\mathbf{U}\|_{\text{F}}^2 - \gamma_{\max}\operatorname{tr}\mathbf{U}\mathbf{U}^\top + c_1. \tag{36}$$

As $\mathbf{U}$ is known, this term can also be combined with c_1 to get constant $\widetilde{c_1} = c_1 - \gamma_{\max}\operatorname{tr}\mathbf{U}\mathbf{U}^\top$. This results in the final expression for $\mathbf{H}$ of

$$L_{\ell_0\text{-SNMF}}(\mathbf{H}|\mathbf{U}) = \gamma_{\max}\|\mathbf{H}-\mathbf{U}\|_{\text{F}}^2 + \widetilde{c_1} \tag{37}$$

$$= \gamma_{\max}\sum_{j=1}^{n}\|\mathbf{h}_j - \mathbf{u}_j\|_{\text{F}}^2 + \widetilde{c_1} \tag{38}$$

From the above equation, we conclude that the loss function can form an error matrix for rows of $\mathbf{H}$. For the ℓ_0-(pseudo)norm constraint, we further need that every row of $\mathbf{H}$ should only have a maximal of F nonzero values. This can be seen as choosing the best subset per row of $\mathbf{U}$, where the best subset is defined as those $\mathbf{u}_j$'s corresponding to the F largest values, as defined by Xiong (2013) for the OLS setting, since then the smallest F errors are retained.

This optimization problem thus simplifies to a sorting problem. First, the $\mathbf{u}_j$'s are sorted in ascending order. The first F elements are then kept, whilst the rest are equated to zero. A problem may arise when a negative value of u_{js} results in the element h_{js} also becoming negative. Nonetheless, an easy solution is to only let an element u_{js} participate in the selection from the largest errors when u_{js} is larger or equal to zero.

More formally, for the vector $\mathbf{x} = (x_1,\ldots,x_r)^\top \in \mathbb{R}^r$, let $\mathcal{U}$ be the set of indices corresponding to the F largest positive values of x_j. We can define the mapping function

$M_F(\mathbf{x}) = \mathbf{z} = (z_1, \ldots, z_r)^\top$, such that

$$M_F(x_j) = z_j = \begin{cases} x_j, & \text{if } j \in \mathcal{U}, \\ 0, & \text{otherwise.} \end{cases} \tag{39}$$

As this function can map the complete result of the rows of $\mathbf{H}$ to the best subset, we define the next iteration as

$$\mathbf{H}^{(k+1)} = M_F(\mathbf{U}) = M_F\left(\mathbf{H}^{(k)} - \frac{1}{\gamma_{\max}}\mathbf{H}^{(k)}\mathbf{W}^\top\mathbf{W} + \frac{1}{\gamma_{\max}}\mathbf{A}^\top\mathbf{W}\right). \tag{40}$$

It is also possible to derive such an expression for ℓ_0-Sparse Weighted Nonnegative Matrix Factorization (ℓ_0-WNMF), as well as ℓ_0-Sparse Graph Regularized Weighted Nonnegative Matrix Factorization (ℓ_0-SGWNMF). The derivations of these can be found in Appendix A.1 and A.2, respectively. To our knowledge, there are no other algorithms capable of simultaneously enforcing ℓ_0-sparsity whilst being robust weighted and graph regularized. Since this makes it difficult to compare our algorithm to any other benchmark, we will not test the performance of these algorithms and further research is needed to determine performance and limitations of these algorithms.

4.3 Implementation

The three ℓ_0-sparse expressions that cover alternative loss functions for NMF, are implemented into the sorting part of the algorithm in a similar manner. In Algorithm 4, the complete procedure for enforcing ℓ_0-sparsity in NMF is shown. In line 9 we need to use the active-set NNLS algorithm on rows of $\mathbf{W}$, as explained in Algorithm 2. We have chosen this method instead of multiplicative updating, because not only does it give optimal results of the subproblem, but the ℓ_0-sparsity enforcing benchmark by Peharz and Pernkopf (2012) also uses NNLS. This makes for a better performance comparison. For the implementation of NNLS in `Matlab`, we use the `lsqnonneg()` function per rows of $\mathbf{W}$ and $\mathbf{H}$.

For the ℓ_0-sparse updating of $\mathbf{H}$, we use the steps described after (38). These steps are the same for all alternative versions of the algorithm below, and only $\gamma_{\max}$ and $\mathbf{U}$ vary per version. For the update, it is first necessary to find the largest eigenvalue for every iteration. As the update depends heavily on this largest eigenvalue, we calculate the exact values. For the implementation in `Matlab`, the function `eig()` is used, which returns all eigenvalues. Afterwards, all positive elements of the rows of $\mathbf{U}$ are sorted, for which we use the `sort()` function. Finally, the update of $\mathbf{H}$ is obtained through the mapping function $M_F(\mathbf{U})$.

Algorithm 4: ℓ_0-SNMF, ℓ_0-SWNMF and ℓ_0-SGWNM

Initialization Set $k = 0$
Set $\mathbf{W}^{(0)}$ as a random initialization matrix with $w_{is} \geq 0$ and rank r
Set $\mathbf{H}^{(0)}$ as a random initialization matrix with with $h_{js} \geq 0$ and rank r
Set number of nonzero entries F
If weighted, determine $\mathbf{R}$ and $\mathbf{D_m}$
If graph regularized, determine λ and $\mathbf{L}$
while $L\left(\mathbf{W}^{(k)}, \mathbf{H}^{(k)}\right) - L\left(\mathbf{W}^{(k+1)}, \mathbf{H}^{(k+1)}\right) < \epsilon L\left(\mathbf{W}^{(k+1)}, \mathbf{H}^{(k+1)}\right)$ **do**
    Fix $\mathbf{H}^{(k)}$
    $\mathbf{W}^{(k+1)}$ via Algorithm 2: Active-set NNLS
    Fix $\mathbf{W}^{(k+1)}$
    $\mathbf{H}^{(k+1)}$ via Update rule (40), (59) or (65) depending on the loss function
    Set $k = k + 1$
end
Return $\mathbf{W}^{+} = \mathbf{W}^{(k+1)}$ and $\mathbf{H}^{+} = \mathbf{H}^{(k+1)}$

4.4 Computational Complexity

Since we have the final algorithm for ℓ_0-SNMF, we can determine the computational complexity and compare it to the classic NMF algorithm. Since the big O notation is often used to express computational complexity, it will be used here as well. It must be noted that we assume serial processing.

The computational complexity of classic NMF depends on the computational complexity of the active-set NNLS algorithm by Lawson and Hanson (1995). The most computational expensive computation of NNLS is the calculation of the Moore-Penrose inverse of the input matrix $\mathbf{C}$ of the passive set $\mathcal{P}$, which is at most a $p \times v$ matrix. By using the Gauss-Jordan elimination to determine this exact (pseudo)inverse it takes $O(pv^2)$ for the matrix multiplication and $O(v^3)$ for the decomposition. If $p >> v$, then the term $O(pv^2)$ will dominate. This needs to be performed at most v times until the active set $\mathcal{Z}$ is empty in the outer while loop. Total asymptotic complexity is then $O(pv^3)$. For NMF we have that this matrix is either $\mathbf{W}$ or $\mathbf{H}$ both with dimensions $n \times K$, where n denotes the number of nodes and K the number of communities. As such, the computation time is $O(nK^3)$. However, since we update per rows of $\mathbf{W}$ and $\mathbf{H}$, we need to repeat this n times. The computation time then becomes $O(n^2K^3)$.

The difference between ℓ_0-SNMF and NMF is the method in which the weighting matrix $\mathbf{H}$ is updated. The update for $\mathbf{W}$ can be determined from the computational complexity of classic NMF. In case of using the NNLS update, it is equal to $O(n^2K^3)$. As such, the overall computation cost for updating $\mathbf{W}$ does not change. For the iteration of $\mathbf{H}$, we first need to determine $\mathbf{U}$ from (34), which uses the largest eigenvalue $\gamma_{\max}$ of matrix $\mathbf{W}^\top\mathbf{W}$. Determining this largest eigenvalue will give the most computational cost and is at worst $O(K^3)$. To subsequently determine $\mathbf{U}$, we only use matrix multiplication. Using schoolbook matrix multiplication, this will be at worst $O(n^2K)$. This happens when, for example, the $n \times n$ matrix $\mathbf{A}$ is multiplied by the $n \times K$ matrix $\mathbf{W}$. The algorithm afterwards performs

a row-wise sorting of all positive elements of $\mathbf{U}$. First, we need to determine if elements in $\mathbf{U}$ are positive, which gives nK elements to compare to zero. This results in a complexity of $O(nK)$. The sorting part takes on average n times $O(K \log K)$ and at worst n times $O(nK^2)$ (JaJa, 2000). The mapping function described in (39) can be viewed as a logical statement, where we need to determine for every element in $\mathbf{U}$ if it is larger or equal than the F^{th} largest value the sorted positive $\mathbf{U}$. This totals to a computational complexity of $O(nK)$. Finally, multiplication is needed to set some values to zero which is at most nK values. If the computational complexity is combined for both updates, the overall computational complexity is the at most equal to classic NMF with NNLS. Nonetheless, this method does not capitalize on the enforced sparsity of matrix $\mathbf{H}$ and resulting sparsity of matrix $\mathbf{W}$. As such, we can improve this computation time by storing the matrix in a sparse format.

For the baseline ℓ_0 constraint algorithm by Peharz and Pernkopf (2012), the only guarantee given in terms of computational complexity is that it is not worse than NNLS with $O(n^2K^3)$.

Table 1: Computational cost per iteration of NMF and ℓ_0-SNMF, which is split into the $\mathbf{W}$ update, the $\mathbf{H}$ update and the total.

	NMF	ℓ_0-SNMF-$\mathbf{W}$	ℓ_0-SNMF-$\mathbf{H}$	ℓ_0-SNMF
overall complexity	$O(n^2K^3)$	$O(n^2K^3)$	$O(n^2K)$	$O(n^2K^3)$

4.5 Evaluation

Evaluation of disjoint community detection can be done with any clustering-based metric. Most of these metrics are based on the confusion matrix of the clusters resulting from the algorithm with the ground-truth ones. In community detection literature, Mutual Information (MI) and Normalized Mutual Information (NMI) are the most widely used metrics. However, since MI ranges from zero, meaning no mutual information between cluster $\mathcal{C}$ and ground-truth cluster $\mathcal{C}'$ and $+\infty$, often NMI is chosen as it is bound between $[0, 1]$. We will, therefore, use NMI as our performance measure. Nonetheless, since NMI is just the normalized version of MI, we will first explain how MI is determined and then how it is normalized.

Mutual information measures the reduction of uncertainty for predicting part of the outcome, given the actual outcome. MI between two sets of resulting communities $\mathcal{C} = \{c'_1, \ldots, c'_K\}$ and $\mathcal{C}' = \{c'_1, \ldots, c'_J\}$ is equal to

$$\mathrm{MI}(\mathcal{C}, \mathcal{C}') = \sum_{k=1}^{|\mathcal{C}|} \sum_{j=1}^{|\mathcal{C}'|} p(c_k, c'_j) \log_2 \left(\frac{p(c_k, c'_j)}{p(c_k)p(c'_j)} \right), \tag{41}$$

where $p(c_k, c'_j)$ is the joint distribution of the two discrete variables. It holds that the higher the outcome, the more information is shared between the communities. MI is a symmetric metric, which means that switching $\mathcal{C}$ for $\mathcal{C}'$ and vice-versa will not affect the score.

Since MI measures reduction in uncertainty and the entropy H is a measure for uncertainty, it is equal to the relative entropy (Walther, 2013; Bao-Gang and Yong, 2008). The entropy of a community detection algorithm $\mathcal{C}$ with communities $c_1, \ldots, c_K$ is calculated

through

$$\mathrm{H}(\mathcal{C}) = -\sum_{k=1}^{|\mathcal{C}|} p(c_k) \log_2 p(c_k), \tag{42}$$

where $\log_2$ is used, since information is stored in bits which are either 0 or 1.

However, to determine both MI (or relative entropy), we still need to define what the above probabilities are. For this, the confusion matrix is used. This gives conditional probabilities of predicting $\mathcal{C}$, given ground-truth $\mathcal{C}'$, so $p(c_k|c'_j)$. The marginal probabilities $p(c'_j)$ are equal to the fraction of nodes in the community c'_j in comparison to the total number of nodes. To obtain the joint probability $p(c_k, c'_j)$ and the other marginal probability $p(c_k)$, we use

$$p(c_k, c'_j) = p(c_k|c'_j)p(c'_j) \tag{43}$$

and

$$p(c_k) = \sum_{j=1}^{|\mathcal{C}'|} p(c_k, c'_j). \tag{44}$$

In community detection literature, normalized mutual information is the arithmetically normalized version of mutual information by the possible entropy of the communities. Since MI is dependent on $\mathcal{C}$ and $\mathcal{C}'$, normalization where equal weight is given to both the entropy of $\mathcal{C}$ and $\mathcal{C}'$ will ensure that detection of unbalanced communities will receive the appropriate performance score. Using the previously defined probabilities, NMI can be determined by

$$\mathrm{NMI}\,(\mathcal{C}, \mathcal{C}') = \frac{2\mathrm{MI}(\mathcal{C}, \mathcal{C}')}{\mathrm{H}(\mathcal{C}) + \mathrm{H}(\mathcal{C}')}. \tag{45}$$

5 Data

We have already described how the new method is implemented in Section 4.3 and the metric used to evaluate performance in Section 4.5. To obtain results of the proposed ℓ_0-SNMF method, both synthetic data and real data is used. In Section 5.1, an explanation is given on the setup for comparison of the algorithms. Afterwards, we describe the experimental set-up in Section 5.2, where we differentiate between simulated networks and real-world networks. Finally, we also discuss the necessary perturbation models to evaluate robustness.

5.1 Comparison Study

Since our method is capable of predetermining the number of nonzero elements F of matrix $\mathbf{H}$, we need to compare it to other NMF methods with similar properties. A distinction can thus be made between performance of algorithms where F is unconstrained and performance for algorithms that enforce ℓ_0-sparsity through this constraint. We utilize the normalized mutual information (NMI), explained in Section 4.5, as the evaluation metric to measure performance.

In the unconstrained setting, the ℓ_0-SNMF algorithm is compared to the classic NMF algorithm. Since we have used non-negative least-squares (NNLS) to define the updating rule for $\mathbf{W}$ in the ℓ_0-SNMF algorithm, we will use this same method for solving the classic NMF problem.

When we do enforce an ℓ_0-constraint, the ℓ_0-SNMF algorithm is also compared to a baseline algorithm which is also capable of setting the number of nonzero elements. From all the alternative sparse NMF algorithms introduced in Section 3.2.3, only the algorithm from Peharz and Pernkopf (2012) can do this. The ℓ_0 constraint NMF algorithm, ℓ_0-cNMF for short, uses the nonnegative constrained orthogonal matching pursuit to approximate the ℓ_0-norm. Additionally, the nonnegative least squares algorithm is used for alternating updating. It must be noted that this method cannot handle the unconstrained NMF loss function, as it needs $F < r$. Peharz and Pernkopf implemented this algorithm in a `Matlab` code which can be found in the following GitHub repository: `https://github.com/smatmo/l0-sparse-NMF`.

So, we compare ℓ_0-SNMF with $F = K$ to NMF and ℓ_0-SNMF with $F = 1$ to ℓ_0-NMF with the same constraint for F. We want to analyze if the performance between the two pairs of comparisons is significantly better or worse. To obtain these performance results for comparison, we use synthetically generated networks as well as real-world networks and compare the NMI of the two pairs of methods. Since noise is also a characteristic of real-world networks, we also want to measure performance in the presence of such contamination.

To analyze statistical significance of the resulting pairs of NMI, either a parametric or non-parametric test can be performed. The former is often done through a paired t-test and the latter through the Wilcoxon Rank-Sum test. The main difference between the two is that the paired t-test assumes normality of the mean difference between the two samples. To make the least number of assumptions about the data, we choose the Wilcoxon Rank-Sum test. The Wilcoxon's rank sum test is in this work performed at a 5% significance level on the (rank of the) NMIs per method from all the repetitions compared to the NMIs of another method.

5.2 Experimental Setup

For the above comparison study, certain network data are necessary. A division is made between simulated data and real-world data. We choose network data which are often used in community detection literature. For the simulated network, one benchmark is used, as discussed in Section 5.2.1. For real-world networks, we gather seven popular real-world networks from Rossi and Ahmed (2015), whilst also choosing networks with different number of nodes, edges and communities. This is further explained in Section 5.2.2. The comparison of performance with noise is done through network data with perturbations. How these perturbations in the network are obtained is explained in Section 5.2.3

All the experiments in this work are performed on a laptop with Intel(R) Core(TM) i5-6300U CPU, 2.40GHz, 8 GB RAM from which 7.85 GB usable.

5.2.1 Synthetic Data

The two most common benchmark networks used are the Girvan Newman (GN) network by Girvan and Newman (2002) and the LFR network by Lancichinetti et al. (2008). The GN network holds 128 nodes partitioned into four equal-sized communities and each node has an expected degree of 16. However, due to the manner in which the GN network is simulated, it has two drawbacks: all nodes have the same expected degree and all communities have equal size. These features are unrealistic, as real-world networks are known to have heterogeneous distributed nodal degree and unbalanced community sizes. We will, therefore, only consider the LFR network.

The LFR network has total number of nodes N, average and maximum degree fixed to respectively 500, 20 and 50 in this setting. The nodes are generated using a power law distribution with parameter τ_1 equal to 3, the aforementioned average and the maximum degree. Following Wu et al. (2018), the degree of edges connecting the different communities with the total degree of edges is measured by μ and varies from 0.1 to 0.8. In particular $\mu \in \{0.1, 0.2, 0.4, 0.8\}$. This means for every ten edges, 10μ are edges that bind two communities together. The community sizes are also generated using a power law distribution with parameter τ_2 equal to 1.5 in this simulation. The advantage of this benchmark is that it introduces both heterogeneity in the degree per node as well as different community sizes, by mixing the communities.

5.2.2 Real Data

The performance of the novel method is also evaluated using real-world data, where noise is much more probable. Since we are not developing an integrated method capable of solving the optimization problem and knowing the number of communities K simultaneously, we assume that K is known beforehand. We conducted experiments on seven real-world networks, with known communities $\mathcal{C}$ and K. We gather these seven open-source real-world networks from Rossi and Ahmed (2015). In Table 2 an overview of some descriptive parameters of the chosen networks is given.

Table 2: Summary of descriptive parameters of seven real networks

Dataset	**|V|**	**|E|**	K	Sparsity	**Author**
Karate	34	78	2	0.865	Girvan and Newman (2002)
Dolphins	62	159	2	0.917	Lusseau and Newman (2004)
Polbooks	105	441	3	0.920	Newman (2006)
Football	115	613	12	0.907	Girvan and Newman (2002)
Polblogs	1,490	16,718	2	0.985	Adamic and Glance (2005)
Cora	2,297	5,429	7	0.999	McCallum et al. (2000)
Citeseer	3,312	4,732	6	0.999	Giles et al. (1998)

5.2.3 Perturbed Data

In this work, we choose two different perturbation models, as introduced by Mitri et al. (2017). These models use random edge perturbation (i.e. addition and deletion), as they do not assume any distribution of the perturbations. It must be noted that when edges are deleted In general, for a graph G and its random graph model $\mathbb{G}(n)$, we model the probability of addition or deletion between two nodes v_i, v_j. This is done through the noise model $\theta(G, \mathbb{G}, \nu_a, \nu_d)$. This probability is equal to:

$$\mathbb{P}_\theta(v_i, v_j) = \begin{cases} \nu_a \mathbb{P}_\mathbb{G}(v_i, v_j) \text{ if } v_i, v_j \notin \mathbb{G}, \\ \nu_d \mathbb{P}_\mathbb{G}(v_i, v_j) \text{ if } v_i, v_j \in \mathbb{G}, \end{cases} \tag{46}$$

where ν_a and ν_d are the probabilities of addition and deletion respectively, and $\mathbb{P}_\mathbb{G}(v_i, v_j)$ is the probability that edge e_{ij} is selected. Hence, if an edge between nodes (v_i, v_j) exists, it is added with probability $\nu_a \mathbb{P}_\mathbb{G}(v_i, v_j)$ and deleted with probability $\nu_d \mathbb{P}_\mathbb{G}(v_i, v_j)$. We choose different levels of noise for addition and deletion, resulting in $\nu_a, \nu_d \in [0, 0.1]$ and repeat this perturbation procedure 200 times. By choosing different probabilities, the following random edge perturbation models are obtained:

Uniform Model Every edge has the same probability of being chosen for addition or deletion, such that $\mathbb{P}_\mathbb{G}(v_i, v_j) = 1/n$. Choosing this uniform probability, results in the Erdös-Rény random graph $\mathbb{G}$ (Erdős and Rényi, 1960). Due to the simplicity of this model, it is easy to determine the direct effect of deletion Mitri et al. (2017). Nonetheless, this model has the limitation that it does not resemble real-world networks, which show strong clustering in the sense that when two nodes have a common neighbour, they have a higher chance of being in the same cluster.

Power law Model This model allows for a closer resemblance to real-world noise. The probability of addition or deletion for edge e_{ij} is proportional to the product of the degrees of vertex v_i and v_j, such that $\mathbb{P}_\mathbb{G}(v_i, v_j) \propto \delta(v_i) \cdot \delta(v_j)$ (Aiello et al., 2001). Thus, the degree sequence follows a power law distribution. Nodes that have a high degree and thus that are important for connectedness have a higher chance of being chosen for altering. Therefore, not only is this type of perturbation model closer to real-world noise, but it also alters the network more significantly.

6 Results

The overall goal is to know how the novel ℓ_0-SNMF algorithm performs and also how it performs in comparison to other baselines. For this, we show in this section the results obtained from following the proposed setup on the aforementioned datasets, where we distinguish between the synthetic LFR network and seven real network datasets. The performance results in terms of accuracy are shown in section 6.1, whilst the performance in terms of speed is presented in Section 6.2. Finally, we perform a sensitivity analysis on the hyperparameter of ℓ_0-SNMF in Section 6.3

6.1 Comparison Study

Since it is common for real-world networks to be sparse, introducing a sparsity constraint should not significantly decrease the accuracy of the community detected from the network. Since our method is capable of updating both with and without ℓ_0-constraint, we test our proposed ℓ_0-SNMF method against two baselines. When the number of nonzero elements is unconstrained, it is classic NMF with NNLS, whilst when this is constrained, this changes to ℓ_0-cNMF by Peharz and Pernkopf (2012).

We compare our proposed method to these baselines for synthetic data in Section 6.1.1. Similarly, we do this for real data in Section 6.1.2. Finally, as we also are interested in the behaviour of our proposed method in the presence of contamination, which we research in Section 6.1.3.

6.1.1 Synthetic Data

We will firstly discuss the results of the course of the average NMIs overall starting values for the varying levels of interconnectedness μ. This is done for both the unconstrained and constrained setting. Afterwards, we zoom in to one μ value, chosen here as 0.1, and show the boxplot of the NMIs per method for all the starting points, as well as a scatterplot, which also shows the dispersion of all the NMIs per method per starting point. Finally, we zoom further into one starting point and we show the loss functions of the methods.

In Figure 3 the NMI of both NMF and ℓ_0-SNMF have nearly the same course as for an increasing amount of interconnected communities, NMI decreases. For the constrained setting the effect of the interconnectedness on the average NMI per method is shown in Figure 4. The ℓ_0-constrained baseline stays lower than the novel ℓ_0-SNMF, throughout the increase of interconnectedness. When $\mu = 0.8$, all constrained and unconstrained methods achieve around the same NMI. This is in line with previous findings, that used LFR as the data generating process (Wu et al., 2018; Cai et al., 2010).

With an interconnectedness of 10%, we firstly compare NMF to ℓ_0-SNMF when there are no constraints on the degrees of freedom, but majorization is still used. In Figure 5 NMF gives similar results to ℓ_0-SNMF. Performing the Wilcoxon sing-rank test at a 5% significance level, the two algorithms do not give significantly different results, as the p-value is equal to 0.63. On the other hand, when adding the ℓ_0-constraint with $F = 1$ to the methods, we get that ℓ_0-SNMF has a significantly higher NMI than its baseline ℓ_0-cNMF, as the p-value is 7.56e-10.

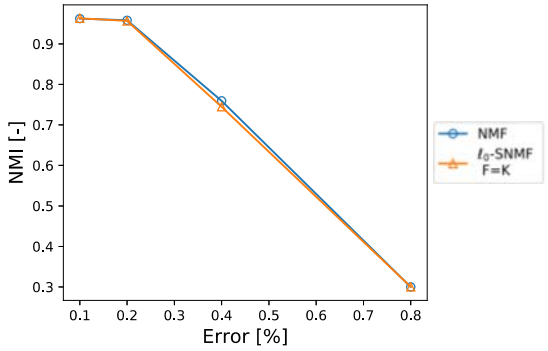


Figure 3: Average NMI when comparing NMF to ℓ_0-SNMF with $F = K$ for uniform perturbed LFR data with μ between 0.1 and 0.8.

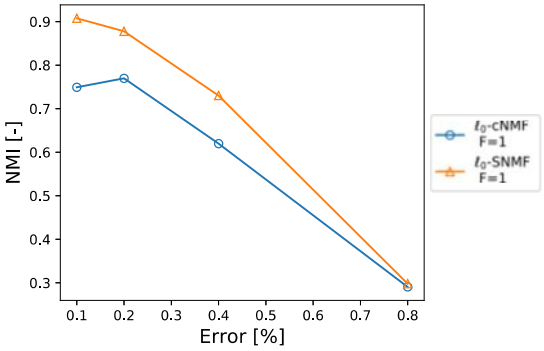


Figure 4: Average NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for uniform perturbed LFR data with μ between 0.1 and 0.8.

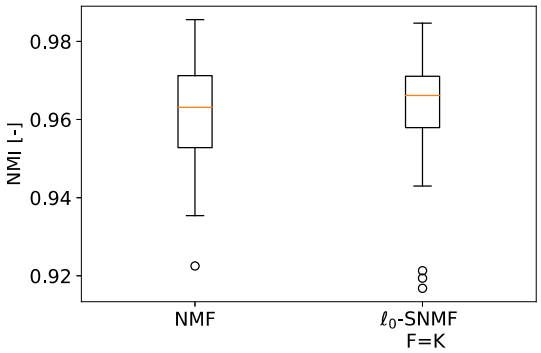


Figure 5: Boxplot of NMI when comparing NMF to ℓ_0-SNMF with $F = K$ for the synthetic dataset $\mu = 0.1$.

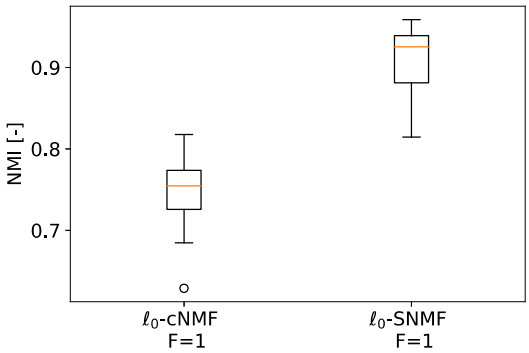


Figure 6: Boxplot of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the synthetic dataset $\mu = 0.1$.

We can also represent the boxplot of all the repetitions as a scatterplot of the NMIs of both methods, with aspect ratio one. For instance in Figure 7, a point in the scatterplot has a y-value equal the NMI of ℓ_0-SNMF and an x-value equal to the NMI of NMF. This means that when both methods have equal NMI it will lie on the black dashed line. When the observations lie above the dashed line, which is the case here, ℓ_0-SNMF has a higher NMI than NMF. Something similar is found in the constrained setting seen in Figure 8 when comparing ℓ_0-SNMF to ℓ_0-cNMF. Although, in this case, the NMI of ℓ_0-SNMF does not remain almost constant and equal to 1.

Finally, to visualize the course of the algorithm's loss function, we normalize it by the squared Frobenius norm of the adjacency matrix and plot it on a log-scaled y-axis. In Figure 9, we see that NMF converges slightly quicker than the novel algorithm and they obtain about the same normalized loss value. For the ℓ_0-constraint algorithms, ℓ_0-cNMF converges the quickest of all algorithms, but the novel algorithm obtains a lower loss than its baseline.

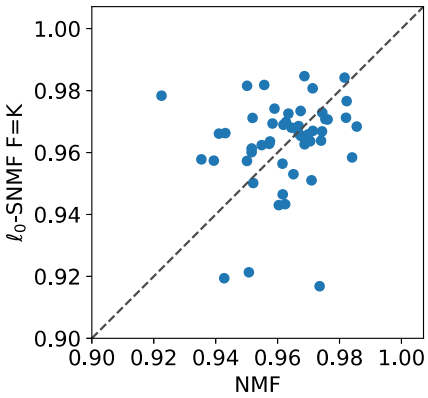


Figure 7: Scatterplot with aspect ratio one of NMI per repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the synthetic dataset $\mu = 0.1$.

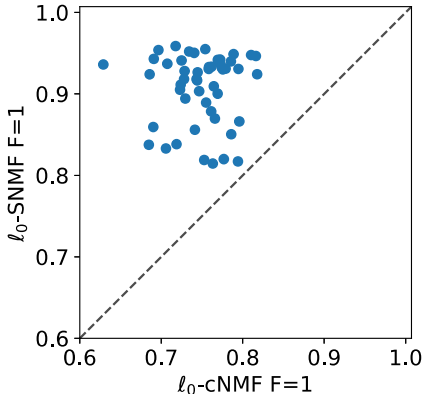


Figure 8: Scatterplot with aspect ratio one of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the synthetic dataset $\mu = 0.1$.

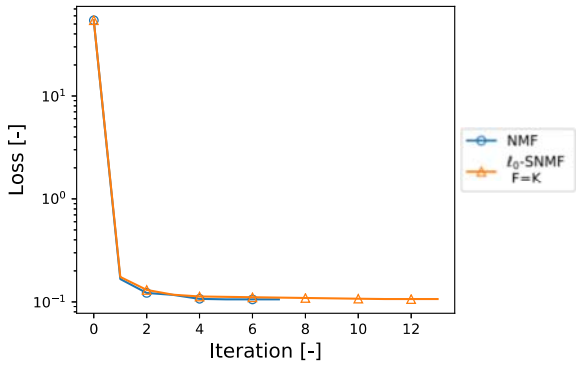


Figure 9: Normalized loss of one repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the synthetic dataset $\mu = 0.1$.

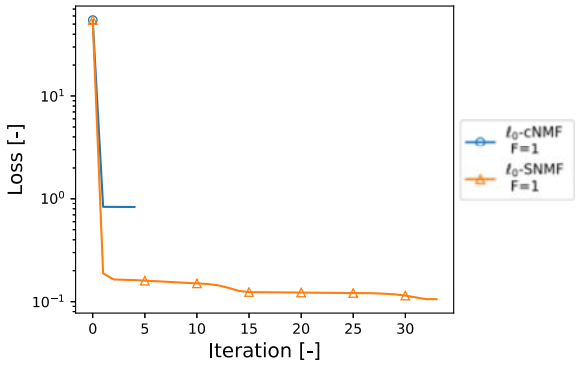


Figure 10: Normalized loss of one repetition when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the synthetic dataset $\mu = 0.1$.

6.1.2 Real Data

Next to synthetic data, it is also of interest to explore how our proposed method performs compared to the benchmarks on real data, as this type of data might be contaminated.

We are interested in when our proposed ℓ_0-SNMF method is superior to ℓ_0-cNMF and if our method performs comparably without the ℓ_0-constraint. When choosing a method, we want both a high NMI and for the method to give stable results in the sense that the NMIs from different starting points are close to each other. The latter means that the standard deviation of the NMIs per method should be small. For the boxplots and scatterplot of the resulting NMI values, as well as the loss for a certain repetition, we refer to Appendix B.2.1, B.2.2 and B.2.3, respectively. The results are summarized in Table 3. For the actual boxplots and scatterplots of these results, Appendix B.2.1 and B.2.2 can be viewed.

Table 3: Summary of results for seven real-world networks in the form of mean (std.dev.), where * denotes significantly better performance between two compared methods at 5% significance level, ** at 1% significance level and *** at 0.1% significance level.

	NMF	ℓ_0-SNMF F = K	ℓ_0-cNMF F =1	ℓ_0-SNMF F =1
Karate	0.773 (0.219)	0.785 (0.196)	0.563 (0.336)*	0.421 (0.253)
Dolphins	0.585 (0.012)	0.588 (0.031)	0.387 (0.110)	0.580 (0.048)***
Polbooks	0.518 (0.027)	0.495 (0.127)	0.339 (0.097)	0.485 (0.055)***
Football	0.890 (0.028)	0.889 (0.033)	0.712 (0.071)*	0.681 (0.083)
Polblogs	0.477 (0.034)	0.461 (0.019)	0.132 (0.111)	0.418 (0.087)***
Cora	0.092 (0.011)	0.090 (0.008)	0.055 (0.014)	0.093 (0.024)**
Citeseer	0.238 (0.083)	0.206 (0.028)	0.070 (0.028)	0.122 (0.030)*

In Table 3 we see that for the unconstrained setting, ℓ_0-SNMF obtains a higher average NMI than NMF for the Karate and Dolphins dataset. For all other datasets, the NMI of NMF is higher. Nonetheless, when performing the Wilcoxon sign-rank test at a 5% significance level, ℓ_0-SNMF does not significantly differ from NMF in terms of NMI. This holds for all datasets.

Constraining the methods through $F = 1$, we see in Table 3 that our proposed method has a higher average NMI than ℓ_0-cNMF for all datasets except the Karate and Football dataset. In most cases it also obtains a lower standard deviation, meaning that for many different starting values the NMI remains high. However, when ℓ_0-cNMF obtains better results than ℓ_0-SNMF, the performance is only significantly different at a 5% level using the Wilcoxon sign rank test, whilst when the opposite occurs ℓ_0-SNMF has significantly higher NMI at often even a 0.1% level. In these cases, we have higher confidence in our novel algorithm performing better than the baseline.

In Appendix B.2.3, the progression of the normalized losses of these algorithms can be found. When we compare these, we see that for the ℓ_0-constraint algorithms, the novel algorithm obtains a lower loss than its baseline for all datasets.

6.1.3 Perturbed Data

Most datasets in real-life applications are perturbed, as contamination might be present. Therefore, it is of interest to see how the proposed ℓ_0-SNMF method, as well as NMF and ℓ_0-cNMF, perform when increasing levels of errors are added.

Uniform Model In case the error is uniformly added to the synthetic LFR dataset with $\mu = 0.1$, we see in Figure 11 that the NMI of NMF and ℓ_0-SNMF decrease. When adding the $F = 1$ constraint we see in Figure 12 that the NMI of ℓ_0-cNMF drops when contaminating the dataset, whereas that of ℓ_0-SNMF remains relatively constant. This hints at some level of contamination resistance of ℓ_0-SNMF.

For the real-life Polbooks dataset, we see in Figure 13 that the performance of both NMF and unconstrained ℓ_0-SNMF becomes erratic and decreases when introducing error contamination. For a constrained setting seen in Figure 14, the course of the NMI of ℓ_0-

SNMF and that of ℓ_0-cNMF under an increasing amount of error is roughly equal, again implying some level of contamination resistance. However, the value of the NMI for ℓ_0-SNMF is still higher, also seen in Figure 47.

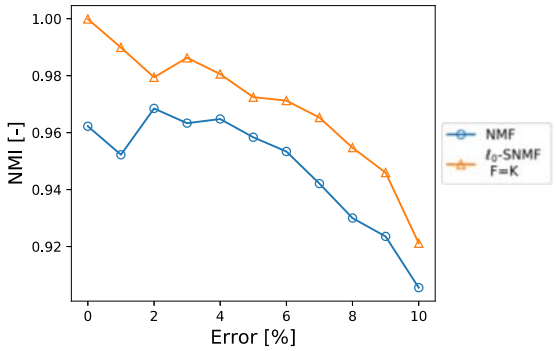


Figure 11: Average NMI when comparing NMF to ℓ_0-SNMF with $F = K$ for uniform perturbed LFR data with $\mu = 0.1$.

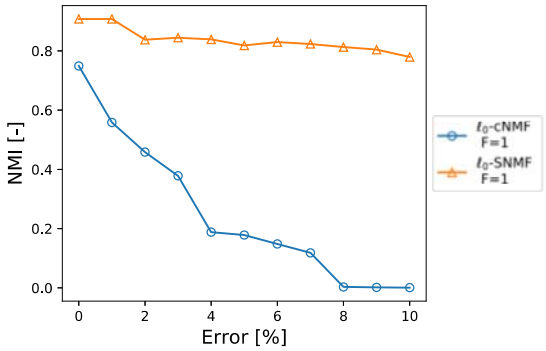


Figure 12: Average NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for uniform perturbed LFR data with $\mu = 0.1$.

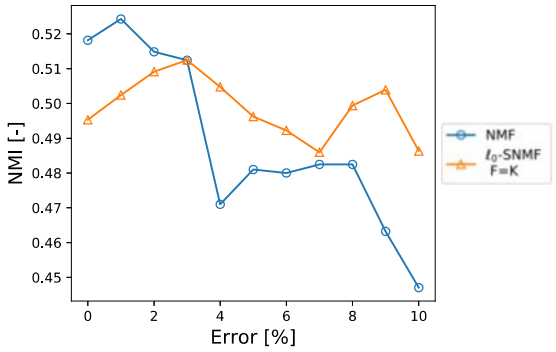


Figure 13: Average NMI when comparing NMF to ℓ_0-SNMF with $F = K$ for uniform perturbed Polbooks data.

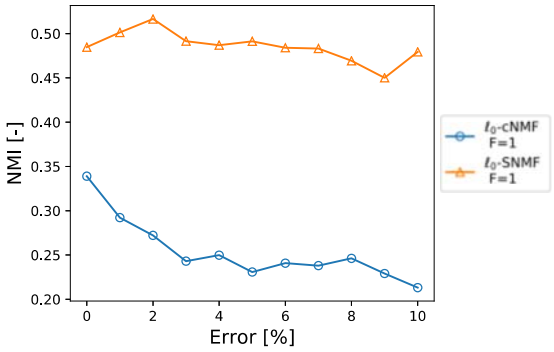


Figure 14: Average NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for uniform perturbed Polbooks data.

Power Law Model For error contamination occurring using the power law we see in Figure 15 for the LFR dataset that the NMI for both NMF and ℓ_0-SNMF when $F = K$ first decrease when introducing error, but after a certain percentage of error the NMI remains relatively constant. Also visible is that NMF performs much worse than ℓ_0-SNMF under power law contamination. Something similar can be seen in Figure 16 when constraining $F = 1$. we see that again ℓ_0-SNMF remains relatively constant with increasing level of contamination, with a higher NMI than ℓ_0-cNMF.

For the Polbooks dataset seen in Figure 17 and 18, ℓ_0-SNMF is again more stable when contamination is present than in both an unconstrained and constrained setting. In the unconstrained setting, we see in Figure 17 that NMF is highly affected by contamination,

heavily worsening the NMI, whilst ℓ_0-SNMF does worsen, but the NMI value remains relatively constant. This persistence against outliers is also present in the constrained setting, whereas ℓ_0-cNMF does decrease from 0.35 to 0.2, as seen in Figure 18.

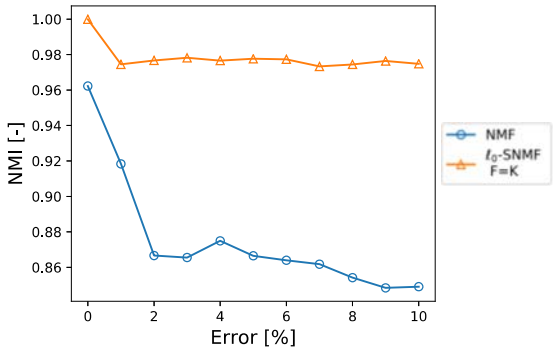


Figure 15: Average NMI when comparing NMF to ℓ_0-SNMF with $F = K$ for Power Law perturbed LFR data with $\mu = 0.1$.

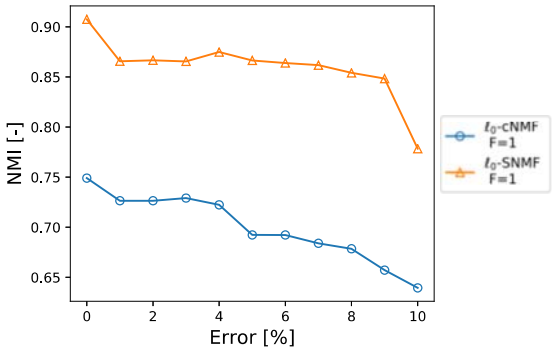


Figure 16: Average NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for Power Law perturbed LFR data with $\mu = 0.1$.

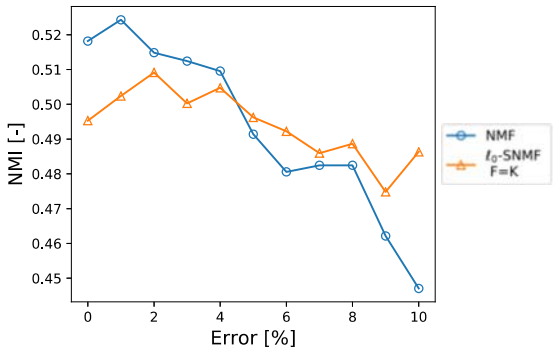


Figure 17: Average NMI when comparing NMF to ℓ_0-SNMF with $F = K$ for Power Law perturbed Polbooks data.

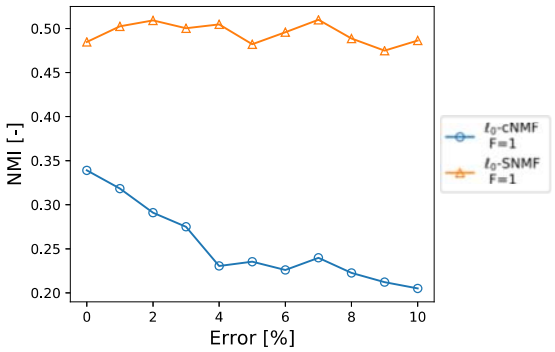


Figure 18: Average NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for Power Law perturbed Polbooks data.

6.2 Computational Complexity

In this section, we show the results of the average time necessary for each algorithm to converge according to the convergence criterion of Algorithm 4. Again, all results were obtained using existing updating algorithms of NMF and ℓ_0-cNMF in MATLAB, as well as our updating algorithm of ℓ_0-SNMF. They are all solved using the existing optimization procedure of NNLS.

In Figure 19 we see that when there is no constraint present on the number of nonzero communities, ℓ_0-SNMF performs better in terms of speed compared to NMF for any number of nodes N ranging from 0 to 3500. This is in line with the theoretical computational time of ℓ_0-SNMF and NMF solved using NNLS. Furthermore, when there are constraints present, ℓ_0-SNMF noticeably outperforms its baseline ℓ_0-cNMF in Figure 20. Starting from around 500 nodes ℓ_0-SNMF takes half the time ℓ_0-cNMF does, and more than half the time after around 3000 nodes.

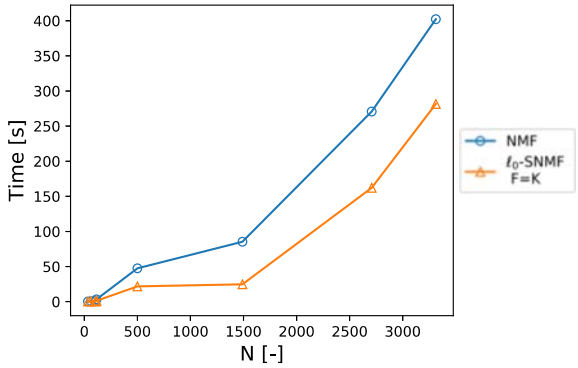


Figure 19: Average time necessary until final convergence of the algorithm against number of nodes when comparing NMF to ℓ_0-SNMF with $F = K$.

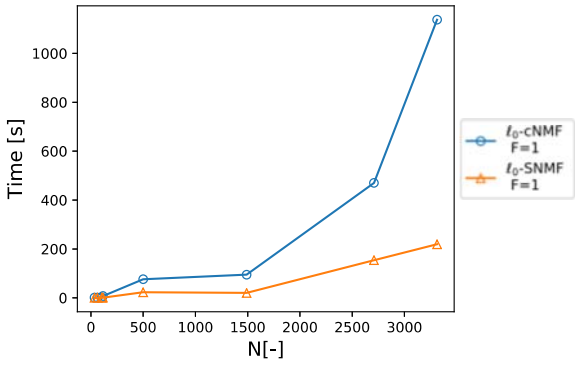


Figure 20: Average time necessary until final convergence of the algorithm against number of nodes when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$.

6.3 Sensitivity Analysis

In this section, we study the effect of the parameter F, which determines the number of nonzero elements per row of matrix $\mathbf{H}$. We have performed a sensitivity analysis on both the synthetic LFR network with $\mu = 0.1$, as well as two real-world networks. We chose the Polbooks network for sensitivity analysis since this dataset is widely used in academic research for sensitivity analysis. We added the Football network, because of its large number of communities. As such, the change in performance from $F = K$ to $F = 1$ for the novel algorithm could be more visible.

From Figure 21, as well as Figure 22a and Figure 22b we see that in all cases ℓ_0-SNMF obtains a higher NMI and also always gives a more constant NMI when increasing the parameter F.

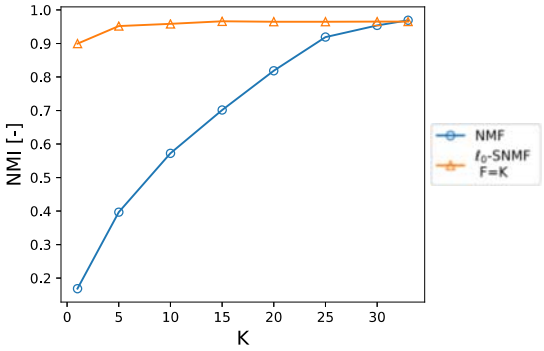
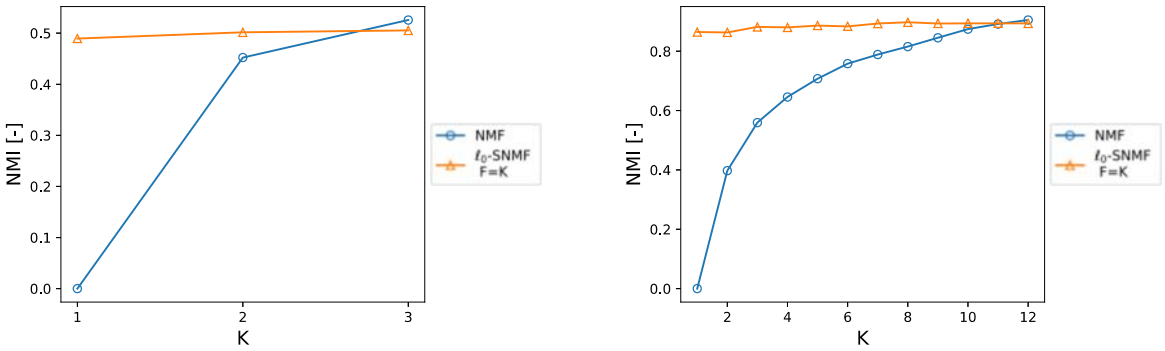


Figure 21: Sensitivity analysis of number of nonzero elements F for ℓ_0-SNMF when compared to classic NMF for multiple values of K and for synthetic LFR network with $\mu = 0.1$.



(a) Polbooks network

(b) Football network

Figure 22: Sensitivity analysis of number of nonzero elements F for ℓ_0-SNMF when compared to classic NMF for multiple values of K and for two real-world networks

6.4 Example: Polbooks

Next to NMI for community detection, the final estimated and ground-truth communities can also be plotted to visualize the differences between the ground-truth and estimated communities. This is done using the spectral positioning representations of the network. This method positions the nodes by the eigenvectors of the graph Laplacian, which serve as the coordinates of the vertices. More important is the confusion matrix of the resulting algorithms, as this can determine the difference between the true node (coloured) labels and the estimated node (coloured) labels more exactly. We choose the Polbooks dataset as it has more than two communities, but does not contain too many nodes. We see that when positioning and visualizing the ground-truth nodes we get Figure 23a for the spectral representation.

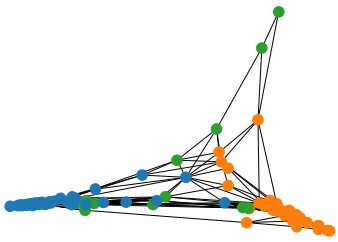
Visualizing the estimated labels from NMF, we find Figure 23b for the spectral representation and to the right of it its confusion matrix. We see that there are quite some nodes that are mislabeled, especially visible in the confusion matrix. A certain node is much more likely to be labelled as a green node, whilst it is instead an orange one.

Comparing the estimated labels of NMF and ℓ_0-SNMF we see that they both give relatively similar results in Figure 23c, in the sense that the nodes that are estimated to be green-labelled are overly present compared to the correct labels, even more so than for NMF. This could be why the average NMI of ℓ_0-SNMF is slightly worse than that of NMF in an unconstrained setting. However, when looking at the confusion matrix, the number of correctly labelled communities is the same as for NMF. But for the smallest community, which is the green one, ℓ_0-SNMF is better at predicting the label.

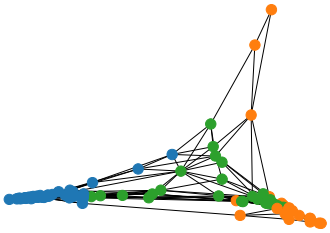
In a constrained setting we see that ℓ_0-cNMF obtains much worse results than the previous few methods, as the estimated green-labelled nodes are overlapping the estimated orange- and blue-labelled nodes, seen in Figure 23d. From the confusion matrix next to it, we see that this algorithm often wrongly predicts a node to belong to the blue community. When looking at the average NMI values and the confusion matrix, ℓ_0-cNMF obtains indeed the worst results.

Finally, constraining the number of non-zero elements per row to 1 with the novel algorithm makes the nodes estimated to be green-labelled much more balanced when compared to its baseline, as seen in Figure 23. From the confusion matrix, we see that green is much less predicted by ℓ_0-SNMF. For this example, the novel algorithm is therefore capable of predicting the unbalanced communities better than the baseline.

Figure 23: Spectral layout of (a) ground-truth communities and spectral layout with confusion matrix for predicted communities of the four algorithms (b-e).

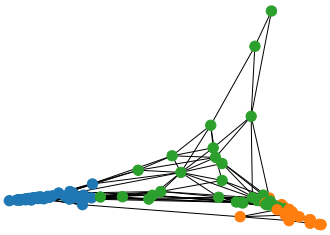


(a) Ground-truth



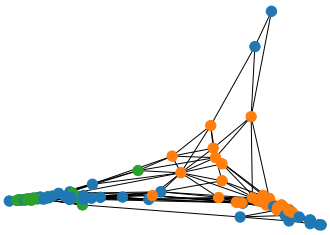
NMF	Ground-Truth		
	Community Blue	Community Orange	Community Green
Predicted Blue	44	0	5
Predicted Orange	0	25	3
Predicted Green	5	18	5

(b) NMF



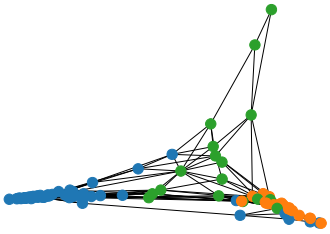
ℓ_0-SNMF $F = K$	Ground-Truth		
	Community Blue	Community Orange	Community Green
Predicted Blue	43	0	5
Predicted Orange	0	25	0
Predicted Green	6	18	8

(c) ℓ_0-SNMF with $F = K$



ℓ_0-cNMF $F = 1$	Ground-Truth		
	Community Blue	Community Orange	Community Green
Predicted Blue	26	13	7
Predicted Orange	3	30	4
Predicted Green	20	0	2

(d) ℓ_0-cNMF with $F = 1$



ℓ_0-SNMF $F = 1$	Ground-Truth		
	Community Blue	Community Orange	Community Green
Predicted Blue	46	5	7
Predicted Orange	0	29	1
Predicted Green	3	9	5

(e) ℓ_0-SNMF with $F = 1$

7 Discussion and Conclusion

In this work, we proposed a novel ℓ_0-Sparse Nonnegative Matrix Factorization (ℓ_0-SNMF) method for the community detection task, which improves upon the NMF with ℓ_0 constraint algorithm by Peharz and Pernkopf (2012). This latter is the baseline method for all Nonnegative Matrix Factorization (NMF) algorithms which use the ℓ_0-(pseudo)norm as a sparsity constraint for NMF. All other sparsity enforcing algorithms cannot directly change the number of nonzero elements of the resulting matrix, since these solely approximate the sparsity in combination with a regularization term.

Iterative majorization was used to obtain a final updating expression, which is capable of enforcing ℓ_0-sparsity, resulting in the best solution. Best in this work was defined as decreasing the loss function of NMF as much as possible. As such, the proposed method set a predefined number of elements equal to zero, if these increase the loss function more than others.

We have implemented and tested our novel algorithm on several datasets. A distinction was made between the synthetic LFR network data by Lancichinetti et al. (2008) and seven real-world networks, which are often used networks to measure the performance of novel community detection algorithms. Additionally, we have shown the effect of contamination on the novel ℓ_0-SNMF algorithm and its accuracy, since noise is a characteristic of real-world networks. The perturbation model by Mitri et al. (2017) was followed, which introduced both uniform and power law perturbations. The latter resembles real-world noise more closely, as densely connected nodes have a higher chance of being perturbed. In all cases, the Normalized Mutual Information (NMI) was used to measure accuracy of the resulting communities when compared to the ground-truth communities.

The results from these studies demonstrate that when comparing the unconstrained ℓ_0-SNMF algorithm with classic NMF, performance is not significantly different. This was concluded using the averages and standard deviations of the NMIs from the methods using multiple starting points and performing the Wilcoxon rank-sum test on the resulting NMI output. Convergence of our proposed method was attained equally fast and computation time was slightly quicker than NMF.

Regarding robustness against perturbations for this comparison, we deduce that the novel ℓ_0-SNMF algorithm without constraint has some resistance against contamination. This was concluded because the NMI remained relatively stable with an increasing level of uniform contamination or power law contamination.

Comparing the novel ℓ_0-SNMF algorithm to the method by Peharz and Pernkopf (2012), we conclude that it performs better both in terms of accuracy and speed for methods where sparsity is high. The novel algorithm obtained communities which are closer to the ground-truth, thus a higher accuracy. Simultaneously, we saw that it was computationally less complex and converged nearly equally fast, improving the performance in terms of speed.

When perturbations were introduced, we conclude that ℓ_0-constrained algorithms, in general, have a more stable performance than unconstrained ones. It is thus also less clear when its performance in terms of accuracy steeply declines. When specifically comparing the two constrained algorithms, we gather that better quality solutions are obtained for the novel algorithm, as its NMI was continuously larger.

Currently, we have tested the unconstrained ℓ_0-SNMF algorithm against NMF with Non-negative Least Squares (NNLS). Nonetheless, there are different algorithms capable of solving the NMF loss function, as described in Section 2.2. As such, it might be possible to obtain different results. Therefore, for future research, it could be of interest to also compare the novel method to other solvers, such as projected gradient descent or multiplicative updating.

Next, when looking at the loss function of all methods for the same repetition, our proposed method always attained a lower loss than ℓ_0-cNMF. This means that the approximation of matrix $\mathbf{A}$ is better when using ℓ_0-SNMF with $F = 1$, then the baseline. It is worth exploring the (possible) implications and applications of this.

Moreover, in this work we have given an expression to enforce ℓ_0-sparsity for rows of the weighting matrix $\mathbf{H}$, which thus lets a node belong only to a limited number of communities. Nonetheless, we could also have applications where only a limited number of nodes are allowed to be in a certain community or one where we want sparse bases vectors for the latent network representations. Combinations of these might also be possible. In this case, we would need to enforce sparsity for columns of $\mathbf{H}$ or $\mathbf{W}$, respectively, or a combination of both rows and columns simultaneously. However, we have not studied the effect of enforcing sparsity on these alternatives, which might be of interest for future research.

Finally, we have applied this algorithm to the community detection task, but NMF is used in other different application tasks, such as computer vision, signal processing and recommender systems. In these applications, the ℓ_0-sparsity constraint could also be of interest. A possible future study could, therefore, study the performance of the novel ℓ_0-SNMF algorithm for other tasks.

References

Adamic, L. A. and Glance, N. (2005), "The political blogosphere and the 2004 US election: divided they blog," in *Proceedings of the 3rd international workshop on Link discovery*, ACM, pp. 36–43.

Aharon, M., Elad, M., and Bruckstein, A. M. (2005), "K-SVD and its non-negative variant for dictionary design," in *Wavelets XI*, International Society for Optics and Photonics, vol. 5914, p. 591411.

Aiello, W., Chung, F., and Lu, L. (2001), "A random graph model for power law graphs," *Experimental Mathematics*, 10, 53–66.

Bai, J. (1995), "Least absolute deviation estimation of a shift," *Econometric Theory*, 11, 403–436.

Bao-Gang, H. and Yong, W. (2008), "Evaluation criteria based on mutual information for classifications including rejected class," *Acta Automatica Sinica*, 34, 1396–1403.

Bijleveld, C. C. and De Leeuw, J. (1991), "Fitting longitudinal reduced-rank regression models by alternating least squares," *Psychometrika*, 56, 433–447.

Bordenave, C., Lelarge, M., and Massoulié, L. (2015), "Non-backtracking spectrum of random graphs: community detection and non-regular ramanujan graphs," in *56th Annual Symposium on Foundations of Computer Science*, Institute of Electrical and Electronics Engineers, pp. 1347–1357.

Cai, D., He, X., Han, J., and Huang, T. S. (2010), "Graph regularized nonnegative matrix factorization for data representation," *Transactions on pattern analysis and machine intelligence*, 33, 1548–1560.

Cai, T. T. and Wang, L. (2011), "Orthogonal matching pursuit for sparse signal recovery with noise," *Transactions on Information Theory*, 57.

Cannistraci, C. V., Alanis-Lobato, G., and Ravasi, T. (2013), "From link-prediction in brain connectomes and protein interactomes to the local-community-paradigm in complex networks," *Scientific reports*, 3, 1613.

Daube-Witherspoon, M. E. and Muehllehner, G. (1986), "An iterative image space reconstruction algorthm suitable for volume ECT," *Transactions on Medical Imaging*, 5, 61–66.

De La Torre, F. and Black, M. J. (2003), "A framework for robust subspace learning," *International Journal of Computer Vision*, 54, 117–142.

De Leeuw, J. (1977), "Applications of convex analysis to multidimensional scaling," *Recent Developments in Statistics*, 133–145.

Ding, C., He, X., and Simon, H. D. (2005), "On the equivalence of nonnegative matrix factor-

ization and spectral clustering," in *Proceedings of the 2005 SIAM International Conference on Data Mining*, SIAM, pp. 606–610.

Donoho, D. L. and Elad, M. (2003), "Optimally sparse representation in general (nonorthogonal) dictionaries via 1 minimization," *Proceedings of the National Academy of Sciences*, 100, 2197–2202.

Du, L., Li, X., and Shen, Y.-D. (2012), "Robust nonnegative matrix factorization via half-quadratic minimization," in *12th International Conference on Data Mining*, Institute of Electrical and Electronics Engineers, pp. 201–210.

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al. (2004), "Least angle regression," *The Annals of statistics*, 32, 407–499.

Erdős, P. and Rényi, A. (1960), "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, 5, 17–60.

Eriksson, A. and van den Hengel, A. (2012), "Efficient Computation of Robust Weighted Low-Rank Matrix Approximations Using the ℓ_1 Norm," *Transactions on Pattern Analysis and Machine Intelligence*, 34, 1681–1690.

Fortunato, S. (2010), "Community detection in graphs," *Physics reports*, 486, 75–174.

Fortunato, S. and Hric, D. (2016), "Community detection in networks: A user guide," *Physics Reports*, 659, 1 – 44, community detection in networks: A user guide.

Giles, C. L., Bollacker, K. D., and Lawrence, S. (1998), "CiteSeer: An Automatic Citation Indexing System." in *ACM DL*, pp. 89–98.

Girvan, M. and Newman, M. E. (2002), "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, 99, 7821–7826.

Groenen, P. J., Giaquinto, P., and Kiers, H. H. (2003), "Weighted majorization algorithms for weighted least squares decomposition models," Tech. rep., Econometric Institute EI 2003-09.

Huber, P. J. (1964), "Robust Estimation of a Location Parameter," *The Annals of Mathematical Statistics*, 73–101.

JaJa, J. (2000), "A perspective on quicksort," *Computing in Science & Engineering*, 2, 43.

Javed, M. A., Younis, M. S., Latif, S., Qadir, J., and Baig, A. (2018), "Community detection in networks: A multidisciplinary review," *Journal of Network and Computer Applications*, 108, 87 – 111.

Kiers, H. A. (1997), "Weighted least squares fitting using ordinary least squares algorithms," *Psychometrika*, 62, 251–266.

Kim, J. and Park, H. (2008), "Sparse nonnegative matrix factorization for clustering," Tech. rep., Georgia Institute of Technology.

Kong, D., Ding, C., and Huang, H. (2011), "Robust nonnegative matrix factorization using l21-norm," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, ACM, pp. 673–682.

Lancichinetti, A., Fortunato, S., and Kertesz, J. (2009), "Detecting the overlapping and hierarchical community structure in complex networks," *New journal of physics*, 11, 033015.

Lancichinetti, A., Fortunato, S., and Radicchi, F. (2008), "Benchmark graphs for testing community detection algorithms," *Physical review E*, 78, 046110.

Laurberg, H. (2008), "Non-negative matrix factorization: Theory and methods," Ph.D. thesis, Ph. D. Thesis.

Lawson, C. L. and Hanson, R. J. (1995), *Solving least squares problems*, vol. 15, Siam.

Lee, D. D. and Seung, H. S. (1999), "Learning the parts of objects by non-negative matrix factorization," *Nature*, 401, 788.

— (2001), "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, pp. 556–562.

Lin, C.-J. (2007), "Projected gradient methods for nonnegative matrix factorization," *Neural computation*, 19, 2756–2779.

Lin, Z., Xu, C., and Zha, H. (2017), "Robust matrix factorization by majorization minimization," *Transactions on Pattern Analysis and Machine Intelligence*, 40, 208–220.

Luo, Y. and Duraiswami, R. (2011), "Efficient parallel nonnegative least squares on multicore architectures," *SIAM Journal on Scientific Computing*, 33, 2848–2863.

Lusseau, D. and Newman, M. E. (2004), "Identifying the role that animals play in their social networks," *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 271, S477–S481.

McCallum, A. K., Nigam, K., Rennie, J., and Seymore, K. (2000), "Automating the construction of internet portals with machine learning," *Information Retrieval*, 3, 127–163.

Mitri, M., Malliaros, F. D., and Vazirgiannis, M. (2017), "Sensitivity of Community Structure to Network Uncertainty," in *Proceedings of the 2017 SIAM International Conference on Data Mining*, SIAM, pp. 345–353.

Morup, M., Madsen, K. H., and Hansen, L. K. (2008), "Approximate ℓ_0 constrained nonnegative matrix and tensor factorization," in *International Symposium on Circuits and Systems*, Institute of Electrical and Electronics Engineers, pp. 1328–1331.

Needell, D. and Vershynin, R. (2009), "Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit," *Foundations of computational mathematics*, 9, 317–334.

Newman, M. E. (2006), "Modularity and community structure in networks," *Proceedings of the national academy of sciences*, 103, 8577–8582.

Paatero, P. and Tapper, U. (1994), "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, 5, 111–126.

Peharz, R. and Pernkopf, F. (2012), "Sparse nonnegative matrix factorization with ℓ_0-constraints," *Neurocomputing*, 80, 38–46.

Rossi, R. A. and Ahmed, N. K. (2015), "The Network Data Repository with Interactive Graph Analytics and Visualization," in *AAAI*.

Shen, Y., Wen, Z., and Zhang, Y. (2014), "Augmented Lagrangian alternating direction method for matrix separation based on low-rank factorization," *Optimization Methods and Software*, 29, 239–263.

Sun, Y., Babu, P., and Palomar, D. P. (2016), "Majorization-minimization algorithms in signal processing, communications, and machine learning," *Transactions on Signal Processing*, 65, 794–816.

Vavasis, S. A. (2009), "On the complexity of nonnegative matrix factorization," *SIAM Journal on Optimization*, 20, 1364–1377.

Walther, D. B. (2013), "Using confusion matrices to estimate mutual information between two categorical measurements," in *2013 International Workshop on Pattern Recognition in Neuroimaging*, Institute of Electrical and Electronics Engineers, pp. 220–224.

Wang, F., Li, T., Wang, X., Zhu, S., and Ding, C. (2011), "Community discovery using nonnegative matrix factorization," *Data Mining and Knowledge Discovery*, 22, 493–521.

Wu, W., Kwong, S., Zhou, Y., Jia, Y., and Gao, W. (2018), "Nonnegative matrix factorization with mixed hypergraph regularization for community detection," *Information Sciences*, 435, 263–281.

Xiong, S. (2013), "Better subset regression," *Biometrika*, 101, 71–84.

Yang, S., Hou, C., Zhang, C., and Wu, Y. (2013), "Robust non-negative matrix factorization via joint sparse and graph regularization for transfer learning," *Neural Computing and Applications*, 23, 541–559.

Zheng, Y., Liu, G., Sugimoto, S., Yan, S., and Okutomi, M. (2012), "Practical low-rank matrix approximation under robust l 1-norm," in *Conference on Computer Vision and Pattern Recognition*, Institute of Electrical and Electronics Engineers, pp. 1410–1417.

Zhou, D., Huang, J., and Schölkopf, B. (2007), "Learning with hypergraphs: Clustering, classification, and embedding," in *Advances in neural information processing systems*, pp. 1601–1608.

Appendices

A Alternative ℓ_0-SNMF Algorithms

In this section, we derive the updating rule of the novel algorithm for NMF with a weighted error loss function and for NMF with a graph regularized weighted error loss function. This is done in Appendix A.1 and A.2, respectively.

A.1 ℓ_0-Sparse Weighted Nonnegative Matrix Factorization (ℓ_0-SWNMF)

To obtain an expression for the updating rule for the ℓ_0-Sparse Weighted NMF (ℓ_0-SWNMF) algorithm, we first need to majorization the weighted loss function. It must be noted, that since we have given a complete explanation on the workings of the algorithm in Section 4.2, we will omit the explanation and only show how the loss function can be rewritten such that sorting is possible.

To do this, we first reformulate NMF as a summation over its residuals e_{ij}, such that

$$L_{\text{WNMF}}\left(\mathbf{H}|\mathbf{A},\mathbf{W},\mathbf{\Omega}\right) = \sum_{j=1}^{n}\sum_{i=1}^{n}\omega_{ij}\left(a_{ij}-\mathbf{w}_i^\top\mathbf{h}_j\right)^2 \tag{47}$$

$$= \sum_{j=1}^{n}\sum_{i=1}^{n}\omega_{ij}e_{ij}^2, \tag{48}$$

where ω_{ij} can be any weight given to the residual. Majorization can be used to rewrite the robust weighted error loss functions from Section 3.2.1 in the form of (48). As such, the weights are derived via majorization and the problem becomes a quadratic loss function. Additionally, these weights can also be known beforehand if the graph itself is weighted.

Following the work of Groenen et al. (2003), which builds upon the work of Kiers (1997), we utilize their row-weighted algorithm for a more efficient and accurate method of solving the weighted least squares loss functions. It is possible to perform weighted majorization row by row, as the loss function can be interpreted as a summation of the loss per row. Following the notation of Groenen et al. (2003), we set $\mathbf{e}$ as the column vectorization of the residuals, such that $\mathbf{e} = \text{vec}(\mathbf{A}-\mathbf{W}\mathbf{H}^\top)$ and $\mathbf{D}_{\mathbf{\Omega}}$ as the $n^2\times n^2$ diagonal matrix with elements ω_{ij}. Then for the row-wise algorithm, we let $\mathbf{e}_i$ denote the column vector of residuals of row i and $\mathbf{D}_{\mathbf{\Omega}_i}$ as the $n\times n$ diagonal matrix of the weights of row i. Then (47) can be rewritten as

$$L_{\text{WNMF}}\left(\mathbf{H}|\mathbf{A},\mathbf{W},\mathbf{D}_{\mathbf{\Omega}}\right) = \mathbf{e}^\top\mathbf{D}_{\mathbf{\Omega}}\mathbf{e} = \sum_{i=1}^{n}\mathbf{e}_i^\top\mathbf{D}_{\mathbf{\Omega}_i}\mathbf{e}_i, \tag{49}$$

which can be majorized using a similar approach as in (29). If we set $\mathbf{m}$ as a vector with elements m_i being the row-maximum of $\mathbf{\Omega}$, then $\mathbf{D}_{\mathbf{\Omega}_i}-m_i\mathbf{I}$ is negative semidefinite. This can be used for finding the following inequality:

$$(\mathbf{e}_i-\mathbf{e}_i^{(k)})^\top(\mathbf{D}_{\mathbf{\Omega}_i}-m_i\mathbf{I})(\mathbf{e}_i-\mathbf{e}_i^{(k)})\leq 0. \tag{50}$$

It again holds that $\mathbf{e}_i^{(k)}$ is chosen as the previous iteration of $\mathbf{e}_i$. By expanding and rearranging this equation, we can majorize the weighted error loss function by

$$L_{\text{WNMF}}\left(\mathbf{H}|\mathbf{A},\mathbf{W},\mathbf{D}_{\mathbf{\Omega}}\right) = \sum_{i=1}^{n}\mathbf{e}_i^\top\mathbf{D}_{\mathbf{\Omega}_i}\mathbf{e}_i$$

$$\leq \sum_{i=1}^{n} m_i \mathbf{e}_i^\top \mathbf{e}_i - 2\sum_{i=1}^{n} m_i \mathbf{e}_i^\top \left[\mathbf{e}_i^{(k)} - \frac{1}{m_i}\mathbf{D}_{\mathbf{\Omega}_i}\mathbf{e}_i^{(k)}\right] + \sum_{i=1}^{n} c_{2i}$$
$$= g_{\text{WNMF}}\left(\mathbf{H}|\mathbf{A}, \mathbf{W}, \mathbf{H}^{(k)}, \mathbf{D}_{\mathbf{\Omega}}\right). \tag{51}$$

If we let $\mathbf{z}_i$ be the part between the square brackets, such that $\mathbf{z}_i = \mathbf{e}_i^{(k)} - \frac{1}{m_i}\mathbf{D}_{\mathbf{\Omega}_i}\mathbf{e}_i^{(k)}$, and we introduce $\mathbf{D}_\mathbf{m}$ as a matrix with diagonal elements equal to $\mathbf{m}$, then we can further simplify the majorization equation to

$$g_{\text{WNMF}}\left(\mathbf{H}|\mathbf{A}, \mathbf{W}, \mathbf{H}^{(k)}, \mathbf{D}_\mathbf{m}\right) = \sum_{i=1}^{n}(\mathbf{e}_i - \mathbf{z}_i)^\top \mathbf{D}_\mathbf{m}(\mathbf{e}_i - \mathbf{z}_i) - \sum_{i=1}^{n} m_i \mathbf{z}_i^\top \mathbf{z}_i + \sum_{i=1}^{n} c_{2i}$$
$$= \operatorname{tr}(\mathbf{E} - \mathbf{Z})^\top \mathbf{D}_\mathbf{m}(\mathbf{E} - \mathbf{Z}) + \operatorname{tr}\mathbf{Z}^\top \mathbf{D}_\mathbf{m}\mathbf{Z} + \sum_{i=1}^{n} c_{2i}, \tag{52}$$

where $\mathbf{Z} = [z_{ij}]_{n\times n}$ is the matrix with elements equal to $z_{ij} = e_{ij}^{(k)} - \frac{\omega_{ij}}{m_i}e_{ij}^{(k)}$. Nonetheless, as for the ℓ_0-sparsity enforcement we need a loss function with $\mathbf{WH}^\top$ stated explicitly, we use the derivation of (17) and (19) in Groenen et al. (2003) to rewrite the first term in (52) to

$$\begin{aligned}
\operatorname{tr}(\mathbf{E} - \mathbf{Z})^\top \mathbf{D}_\mathbf{m}(\mathbf{E} - \mathbf{Z}) &= \sum_{i=1}^{n} m_i \sum_{i=1}^{n}(e_{ij} - z_{ij})^2 = \sum_{i=1}^{n} m_i \sum_{i=1}^{n}\left(e_{ij} - e_{ij}^{(k)} - \frac{\omega_{ij}}{m_i}e_{ij}^{(k)}\right)^2 \\
&= \sum_{i=1}^{n} m_i \sum_{i=1}^{n}\left(a_{ij} - \mathbf{w}_i^\top \mathbf{h}_j - a_{ij}^{(k)} + \mathbf{w}_i^{(k)\top}\mathbf{h}_j^{(k)} - \frac{\omega_{ij}}{m_i}\left(a_{ij}^{(k)} - \mathbf{w}_i^{(k)\top}\mathbf{h}_j^{(k)}\right)\right)^2 \\
&= \sum_{i=1}^{n} m_i \sum_{i=1}^{n}\left(\left[1 - \frac{\omega_{ij}}{m_i}\right]\mathbf{w}_i^{(k)\top}\mathbf{h}_j^{(k)} + \frac{\omega_{ij}}{m_i}a_{ij} - \mathbf{w}_i^\top \mathbf{h}_j\right)^2 \\
&= \sum_{i=1}^{n} m_i \sum_{i=1}^{n}(r_{ij} - \mathbf{w}_i^\top \mathbf{h}_j)^2 \\
&= \operatorname{tr}(\mathbf{R} - \mathbf{WH}^\top)^\top \mathbf{D}_\mathbf{m}(\mathbf{R} - \mathbf{WH}^\top),
\end{aligned} \tag{53}$$

where $r_{ij} = \left[1 - \frac{\omega_{ij}}{m_i}\right]\mathbf{w}_i^{(k)\top}\mathbf{h}_j^{(k)} + \frac{\omega_{ij}}{m_i}a_{ij}$ and $\mathbf{D}_\mathbf{m}$ is an $n \times n$ diagonal matrix with elements m_i. We can then rewrite (52) as

$$\begin{aligned}
g_{\text{WNMF}}(\mathbf{H}|\mathbf{A}, \mathbf{W}, \mathbf{H}^{(k)}, \mathbf{D}_\mathbf{m}) &= \operatorname{tr}(\mathbf{R} - \mathbf{WH}^\top)^\top \mathbf{D}_\mathbf{m}(\mathbf{R} - \mathbf{WH}^\top) + \operatorname{tr}\mathbf{Z}^\top \mathbf{D}_\mathbf{m}\mathbf{Z} + \sum_{i=1}^{n} c_{2i} \\
&= \operatorname{tr}(\mathbf{R} - \mathbf{WH}^\top)^\top \mathbf{D}_\mathbf{m}(\mathbf{R} - \mathbf{WH}^\top) + c_2',
\end{aligned} \tag{54}$$

where we combined the two constant terms into c_2', such that

$$c_2' = \operatorname{tr}\mathbf{Z}^\top \mathbf{D}_\mathbf{m}\mathbf{Z} + \sum_{i=1}^{n} c_{2i}. \tag{55}$$

To obtain an ℓ_0-sparse result, we make use of the same inequality as in (30), but slightly modified to accommodate for the diagonal weighting matrix $\mathbf{D}_\mathbf{m}$. If we let $\gamma_{\max}$ here be the maximum eigenvalue of the matrix $\mathbf{W}^\top \mathbf{D}_\mathbf{m}\mathbf{W}$, then

$$\operatorname{tr}\left(\mathbf{H} - \mathbf{H}^{(k)}\right)\left(\mathbf{W}^\top \mathbf{D}_\mathbf{m}\mathbf{W} - \gamma_{\max}\mathbf{I}\right)\left(\mathbf{H} - \mathbf{H}^{(k)}\right)^\top \leq 0. \tag{56}$$

Since we can expand and rewrite this equation in the exact same manner as before, we can majorize the term $\operatorname{tr}\mathbf{HW}^\top \mathbf{D}_\mathbf{m}\mathbf{WH}^\top$ which results when expanding (54). By following the

same steps as in Section 4.2, we can get a similar expression as in (37), where now

$$\mathbf{U} = \mathbf{H}^{(k)} - \frac{1}{\gamma_{\max}}\mathbf{H}^{(k)}\mathbf{W}^\top\mathbf{D_m}\mathbf{W} + \frac{1}{\gamma_{\max}}\mathbf{D_m}\mathbf{R}^\top\mathbf{W}, \tag{57}$$

and

$$c_1 = \operatorname{tr}\mathbf{R}^\top\mathbf{D_m}\mathbf{R} + \gamma_{\max}\operatorname{tr}\mathbf{H}^{(k)}\mathbf{H}^{(k)\top} - \operatorname{tr}\mathbf{H}^{(k)}\mathbf{W}^\top\mathbf{D_m}\mathbf{W}\mathbf{H}^{(k)\top} + c_2' \tag{58}$$

By then using the mapping function defined in (39), the next iteration becomes

$$\mathbf{H}^{(k+1)} = M_F(\mathbf{U}) = M_F\left(\mathbf{H}^{(k)} - \frac{1}{\gamma_{\max}}\mathbf{H}^{(k)}\mathbf{W}^\top\mathbf{D_m}\mathbf{W} + \frac{1}{\gamma_{\max}}\mathbf{D_m}\mathbf{R}^\top\mathbf{W}\right). \tag{59}$$

Thus, we have an updating rule capable of enforcing ℓ_0-sparsity on the rows of $\mathbf{H}$ for any weighted error loss function. Moreover, with the row-wise majorization function described in (51), we have a more efficient and accurate method of solving any weighted loss function in the form of (47).

A.2 ℓ_0-Sparse Graph Regularized Weighted Nonnegative Matrix Factorization (ℓ_0-SGWNMF)

In Section 3.2.2, we explain that by adding graph regularization to the loss function, a more accurate result is obtained. There has to our knowledge never been an algorithm capable of enforcing ℓ_0-sparsity on loss functions with graph regularization. Therefore, we are interested in deriving a similar expression as in (40) and (59), but including the graph regularization term. This will result in the ℓ_0-Sparse Weighted Graph regularized NMF (ℓ_0-SGWNMF) algorithm. As such, we are not only capable of including the graph regularization, but also any weights that want to be added.

Similar to the other two derivations, we first need to rewrite the loss function. We use the derivation of (54) to rewrite the weighted error loss function into its row-weighted majorized form, such that

$$\begin{aligned} L_{\text{GWNMF}}\left(\mathbf{H}|\mathbf{A},\mathbf{W},\mathbf{\Omega},\lambda\right) &= \sum_{j=1}^{n}\sum_{i=1}^{n}\omega_{ij}\left(a_{ij} - \mathbf{w}_i^\top\mathbf{h}_j\right)^2 + \lambda\operatorname{tr}\mathbf{H}^\top\mathbf{L}\mathbf{H} \\ &\leq \operatorname{tr}(\mathbf{R} - \mathbf{W}\mathbf{H}^\top)^\top\mathbf{D_m}(\mathbf{R} - \mathbf{W}\mathbf{H}^\top) + \lambda\operatorname{tr}\mathbf{H}^\top\mathbf{L}\mathbf{H} + c_2' \\ &= g_{\text{GWNMF}}\left(\mathbf{H}|\mathbf{A},\mathbf{W},\mathbf{H}^{(k)},\mathbf{D_m},\lambda\right) \end{aligned} \tag{60}$$

We can then expand this result to

$$\operatorname{tr}\mathbf{R}\mathbf{D_m}\mathbf{R}^\top + \operatorname{tr}\mathbf{H}\mathbf{W}^\top\mathbf{D_m}\mathbf{W}\mathbf{H}^\top - 2\operatorname{tr}\mathbf{R}\mathbf{D_m}\mathbf{H}\mathbf{W}^\top + \lambda\operatorname{tr}\mathbf{H}^\top\mathbf{L}\mathbf{H} + c_2'. \tag{61}$$

The terms $\operatorname{tr}\mathbf{H}\mathbf{W}^\top\mathbf{D_m}\mathbf{W}\mathbf{H}^\top$ and $\lambda\operatorname{tr}\mathbf{H}^\top\mathbf{L}\mathbf{H}$ make it cumbersome to arrive to a final expression. Nonetheless, we have seen before that when we majorize such terms using (30), we get a majorization function with highest order $\operatorname{tr}\mathbf{H}^\top\mathbf{H}$, which is the same as $\operatorname{tr}\mathbf{H}\mathbf{H}^\top$. Therefore, to overcome the above difficulty, we use two majorization steps one for each term. For the term $\operatorname{tr}\mathbf{H}\mathbf{W}^\top\mathbf{D_m}\mathbf{W}\mathbf{H}^\top$ we use the exact same majorization function as (30), where $\gamma_{\max,1}$ is the maximum eigenvalue of $\mathbf{W}^\top\mathbf{D_m}\mathbf{W}$. For the term $\lambda\operatorname{tr}\mathbf{H}^\top\mathbf{L}\mathbf{H}$ we take the largest eigenvalue $\gamma_{\max,2}$ of $\lambda\mathbf{L}$, which results in

$$\operatorname{tr}\left(\mathbf{H} - \mathbf{H}^{(k)}\right)\left(\lambda\mathbf{L} - \gamma_{\max,2}\mathbf{I}\right)\left(\mathbf{H} - \mathbf{H}^{(k)}\right)^\top \leq 0. \tag{62}$$

Sine we have two majorization steps that give a term $\operatorname{tr}\mathbf{H}\mathbf{H}^\top$, one with $\gamma_{\max,1}$ and the other

with $\gamma_{\max,2}$, we combine them into $(\gamma_{\max,1} + \gamma_{\max,2}) \operatorname{tr} \mathbf{H}\mathbf{H}^\top$. For ease of notation we let $c_3 = \gamma_{\max,1} + \gamma_{\max,2}$.

As in the cases before, $\mathbf{H}$ is the only unknown in the majorized loss function. Therefore, we can combine like terms for $\mathbf{H}$ and join all known terms into constant c_1. By following the same steps as in Section 4.2, we can get a similar expression as in (37), where

$$\mathbf{U} = \mathbf{H}^{(k)} - \frac{1}{c_3}\mathbf{H}^{(k)}\mathbf{W}^\top\mathbf{D_m}\mathbf{W} - \frac{1}{c_3}\lambda\mathbf{L}^\top\mathbf{H} + \frac{1}{c_3}\mathbf{D_m}\mathbf{R}^\top\mathbf{W} \tag{63}$$

and

$$c_1 = \operatorname{tr}\mathbf{R}^\top\mathbf{D_m}\mathbf{R} + c_3 \operatorname{tr}\mathbf{H}^{(k)}\mathbf{H}^{(k)\top} - \operatorname{tr}\mathbf{H}^{(k)}\mathbf{W}^\top\mathbf{D_m}\mathbf{W}\mathbf{H}^{(k)\top} - \lambda \operatorname{tr}\mathbf{H}^{(k)\top}\mathbf{L}\mathbf{H}^{(k)} + c_2'. \tag{64}$$

With the mapping function, the next iteration becomes

$$\mathbf{H}^{(k+1)} = M_F(\mathbf{U}) = M_F\left(\mathbf{H}^{(k)} - \frac{1}{c_3}\mathbf{H}^{(k)}\mathbf{W}^\top\mathbf{D_m}\mathbf{W} - \frac{1}{c_3}\lambda\mathbf{L}^\top\mathbf{H} + \frac{1}{c_3}\mathbf{D_m}\mathbf{R}^\top\mathbf{W}\right). \tag{65}$$

Thus, a similar updating rule can be found to enforce ℓ_0-sparsity for any sort of graph regularized weighted error loss function.

B Additional Results

B.1 Synthetic Data

B.1.1 Boxplots

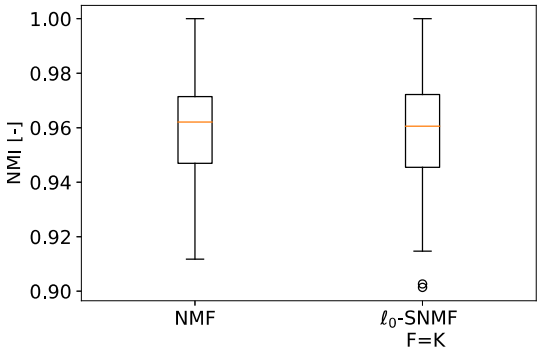


Figure 24: Boxplot of NMI when comparing NMF to ℓ_0-SNMF with $F = K$ for the synthetic dataset with $\mu = 0.2$.

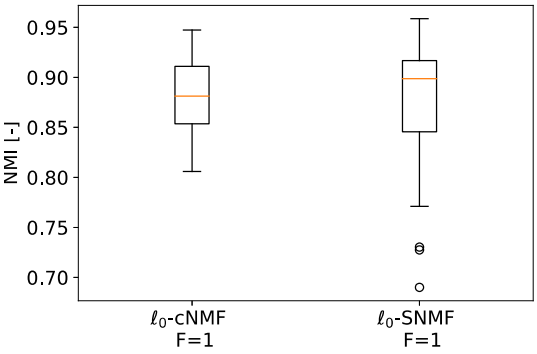


Figure 25: Boxplot of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the synthetic dataset with $\mu = 0.2$.

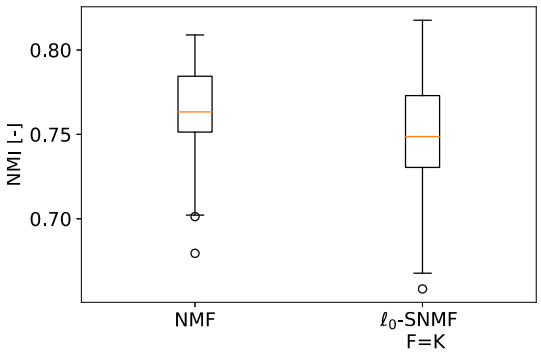


Figure 26: Boxplot of NMI when comparing NMF to ℓ_0-SNMF with $F = K$ for the synthetic dataset with $\mu = 0.4$.

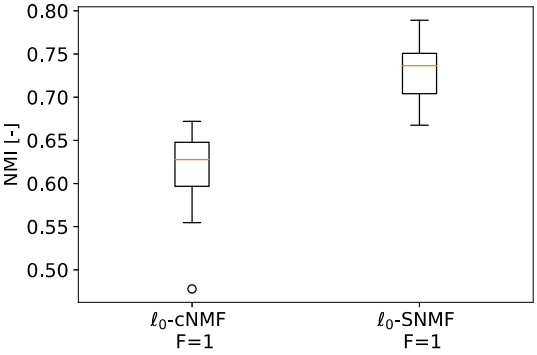


Figure 27: Boxplot of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the synthetic dataset with $\mu = 0.4$.

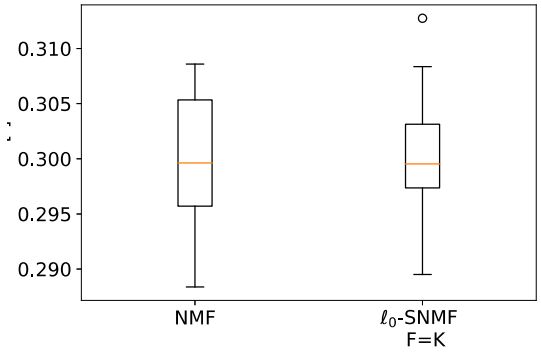


Figure 28: Boxplot of NMI when comparing NMF to ℓ_0-SNMF with $F = K$ for the synthetic dataset with $\mu = 0.8$.

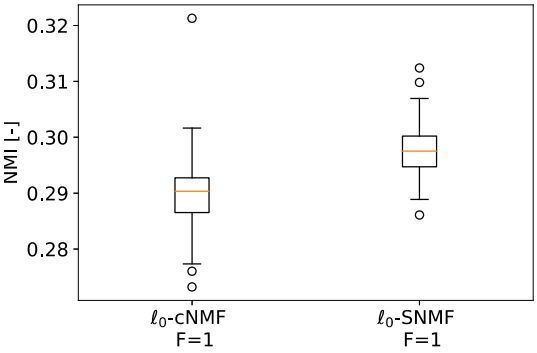


Figure 29: Boxplot of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the synthetic dataset with $\mu = 0.8$.

B.1.2 Scatterplots

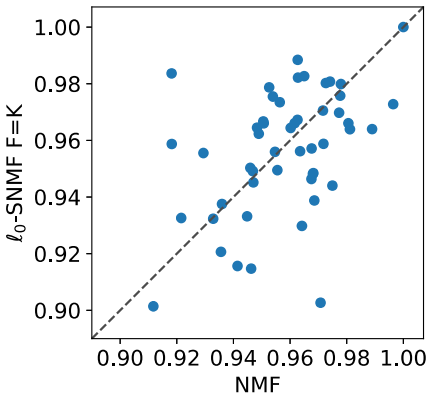


Figure 30: Scatterplot with aspect ratio one of NMI per repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the synthetic dataset with $\mu = 0.2$.

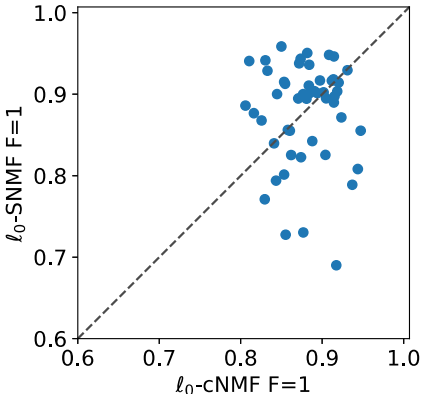


Figure 31: Scatterplot with aspect ratio one of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the synthetic dataset with $\mu = 0.2$.

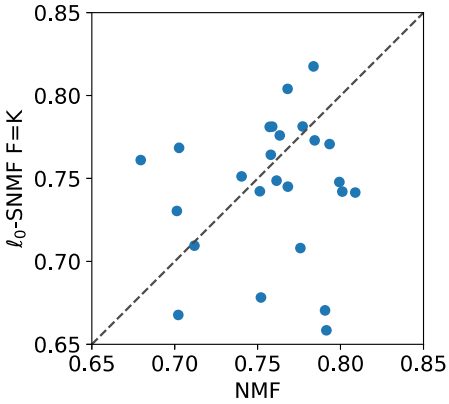


Figure 32: Scatterplot with aspect ratio one of NMI per repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the synthetic dataset with $\mu = 0.4$.

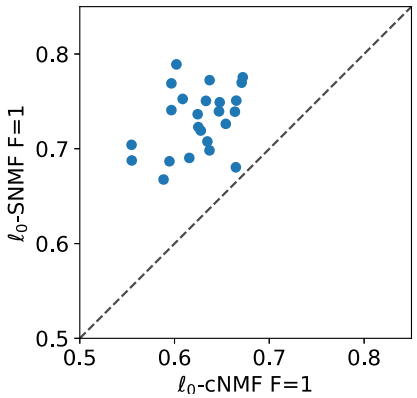


Figure 33: Scatterplot with aspect ratio one of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the synthetic dataset with $\mu = 0.4$.

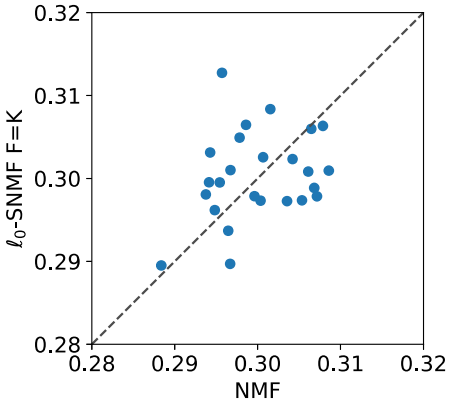


Figure 34: Scatterplot with aspect ratio one of NMI per repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the synthetic dataset with $\mu = 0.8$.

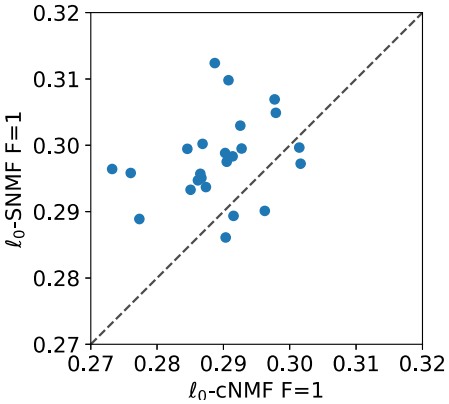


Figure 35: Scatterplot with aspect ratio one of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the synthetic dataset with $\mu = 0.8$.

B.1.3 Normalized loss

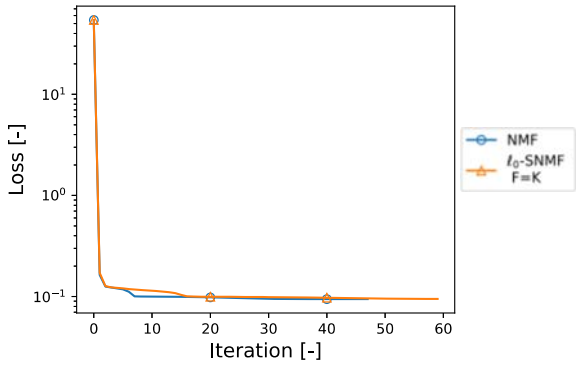


Figure 36: Normalized loss of one repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the synthetic dataset with $\mu = 0.2$.

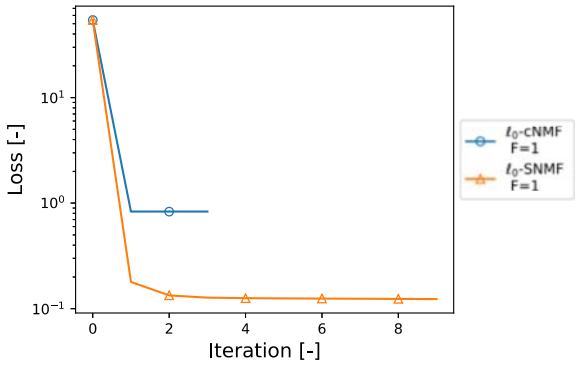


Figure 37: Normalized loss of one repetition when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the synthetic dataset with $\mu = 0.2$.

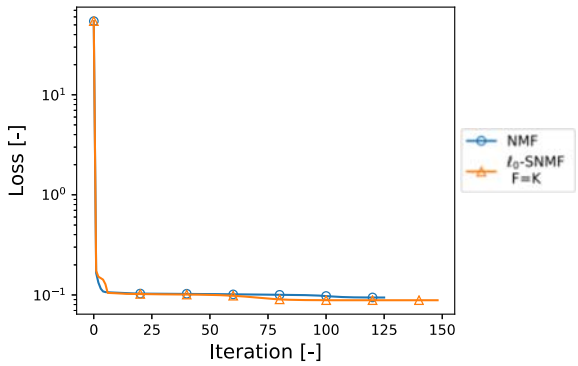


Figure 38: Normalized loss of one repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the synthetic dataset with $\mu = 0.4$.

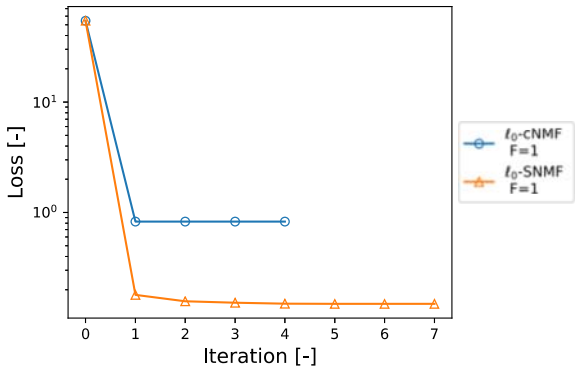


Figure 39: Normalized loss of one repetition when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the synthetic dataset with $\mu = 0.4$.

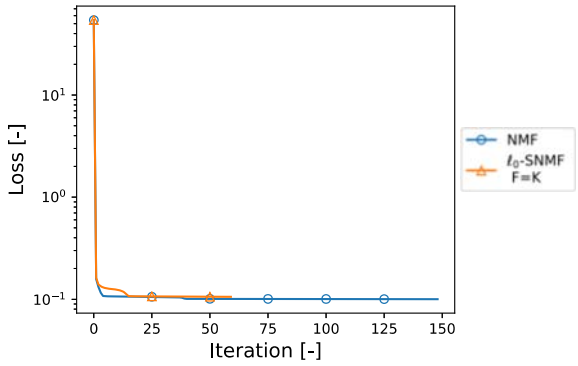


Figure 40: Normalized loss of one repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the synthetic dataset with $\mu = 0.8$.

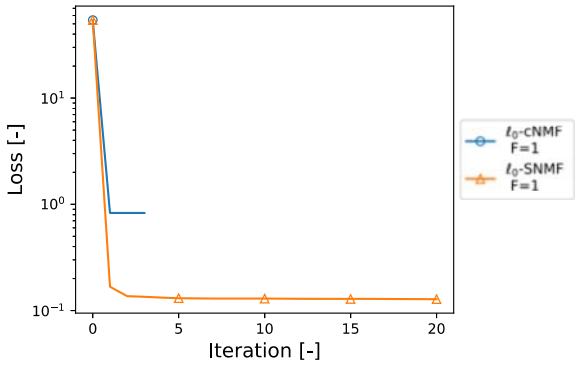


Figure 41: Normalized loss of one repetition when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the synthetic dataset with $\mu = 0.8$.

B.2 Real Data

B.2.1 Boxplots

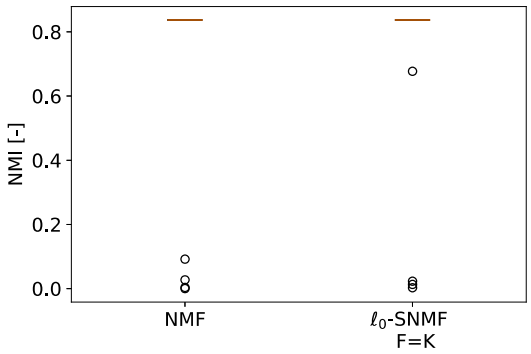


Figure 42: Boxplot of NMI when comparing NMF to ℓ_0-SNMF with $F = K$ for the Karate dataset.

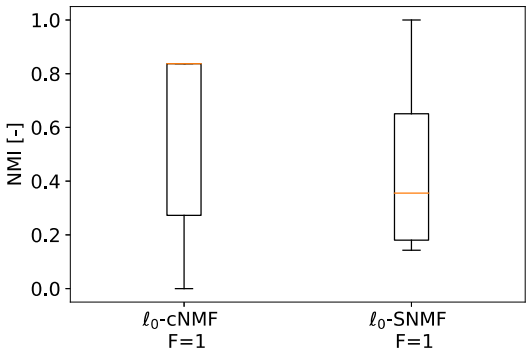


Figure 43: Boxplot of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Karate dataset.

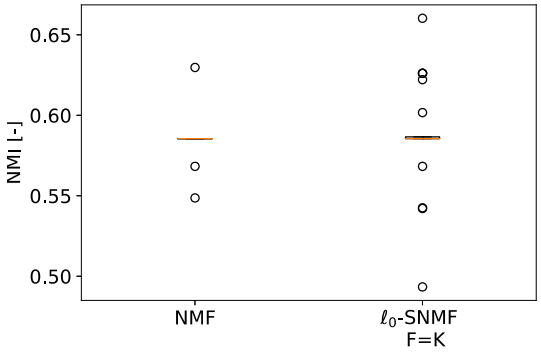


Figure 44: Boxplot of NMI when comparing NMF to ℓ_0-SNMF with $F = K$ for the Dolphins dataset.

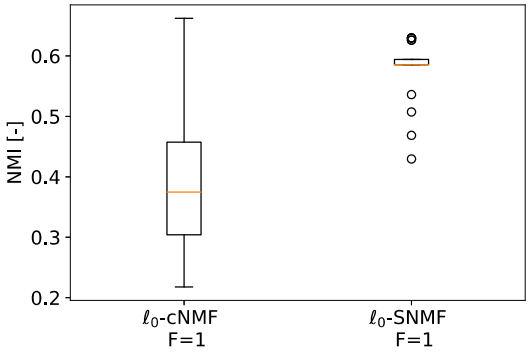


Figure 45: Boxplot of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Dolphins dataset.

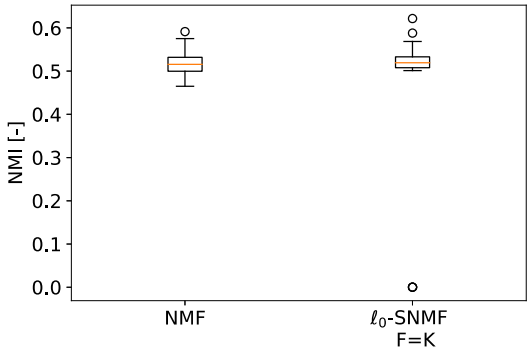


Figure 46: Boxplot of NMI when comparing NMF to ℓ_0-SNMF with $F = K$ for the Polbooks dataset.

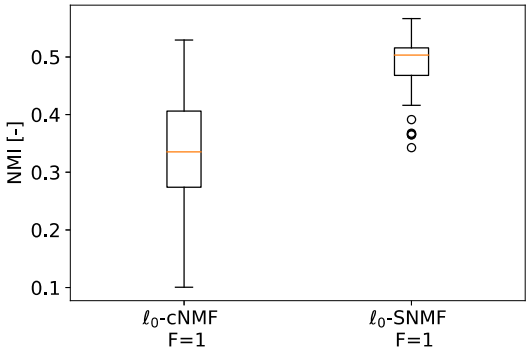


Figure 47: Boxplot of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Polbooks dataset.

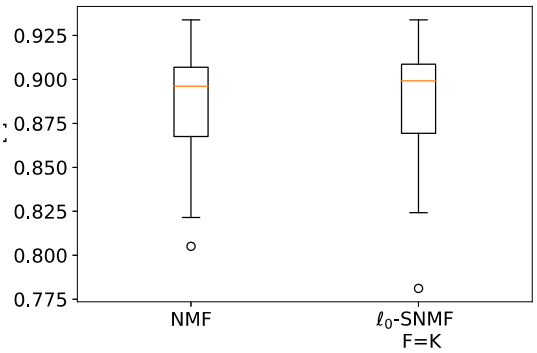


Figure 48: Boxplot of NMI when comparing NMF to ℓ_0-SNMF with $F = K$ for the Football dataset.

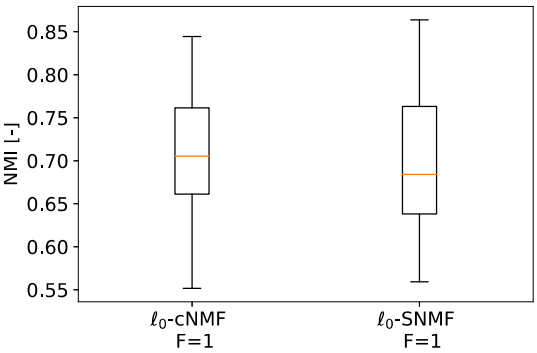


Figure 49: Boxplot of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Football dataset.

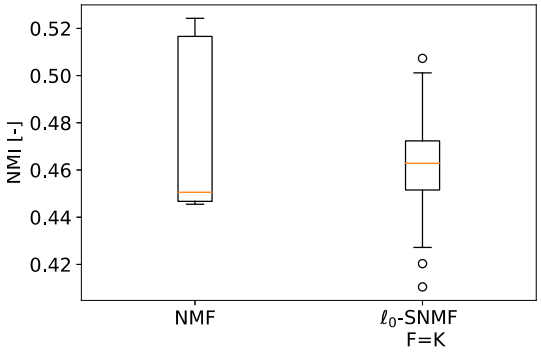


Figure 50: Boxplot of NMI when comparing NMF to ℓ_0-SNMF with $F = K$ for the Polblogs dataset.

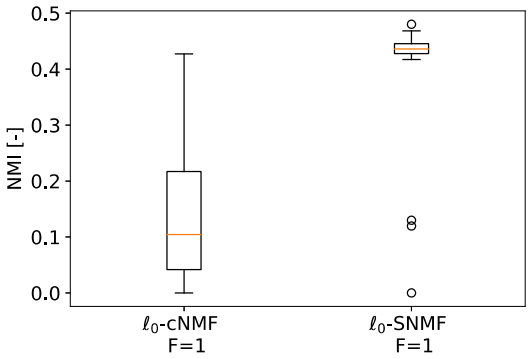


Figure 51: Boxplot of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Polblogs dataset.

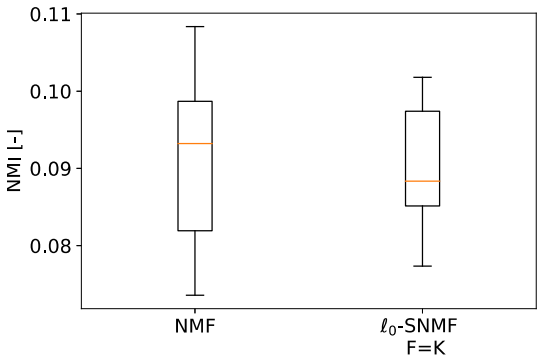


Figure 52: Boxplot of NMI when comparing NMF to ℓ_0-SNMF with $F = K$ for the Cora dataset.

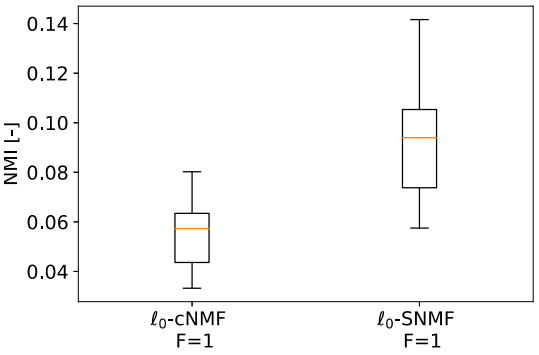


Figure 53: Boxplot of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Cora dataset.

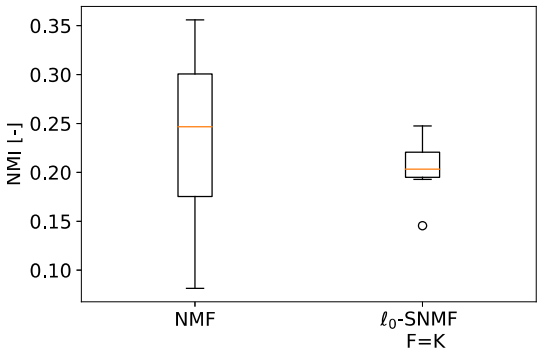


Figure 54: Boxplot of NMI when comparing NMF to ℓ_0-SNMF with $F = K$ for the Citeseer dataset.

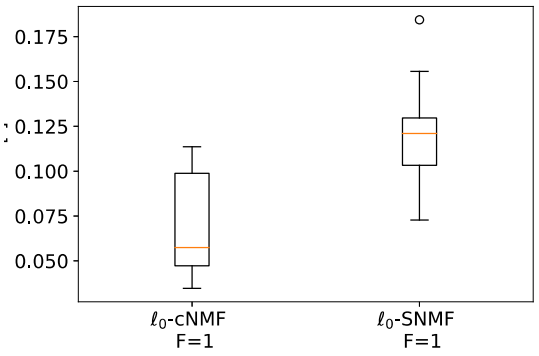


Figure 55: Boxplot of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Citeseer dataset.

B.2.2 Scatterplots

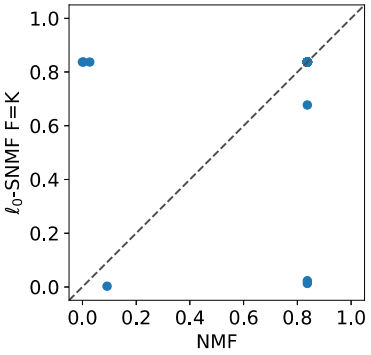


Figure 56: Scatterplot with aspect ratio one of NMI per repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the Karate dataset.

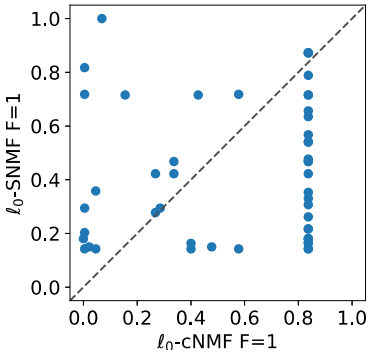


Figure 57: Scatterplot with aspect ratio one of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Karate dataset.

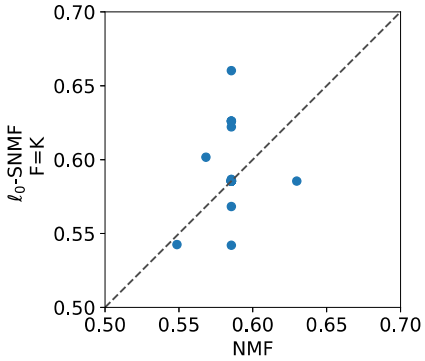


Figure 58: Scatterplot with aspect ratio one of NMI per repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the Dolphins dataset.

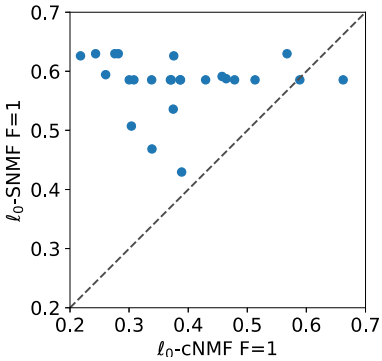


Figure 59: Scatterplot with aspect ratio one of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Dolphins dataset.

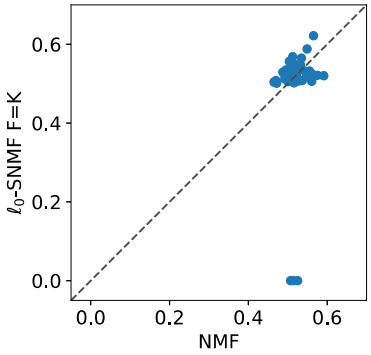


Figure 60: Scatterplot with aspect ratio one of NMI per repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the Polbooks dataset.

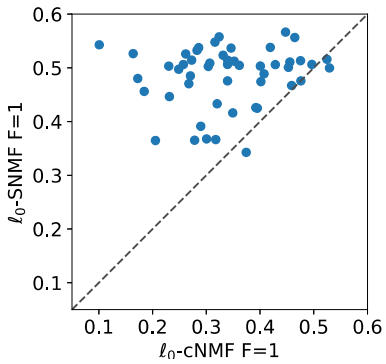


Figure 61: Scatterplot with aspect ratio one of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Polbooks dataset.

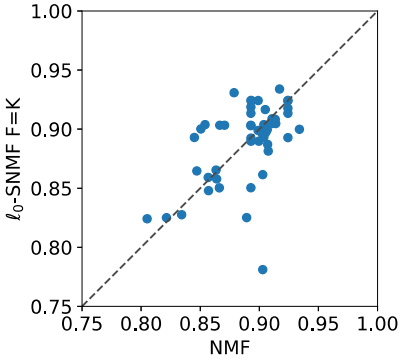


Figure 62: Scatterplot with aspect ratio one of NMI per repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the Football dataset.

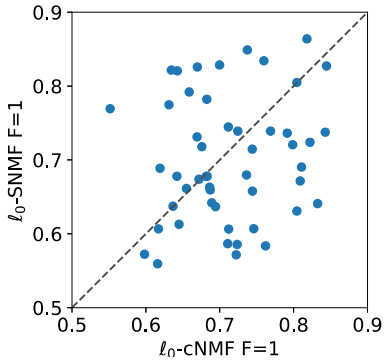


Figure 63: Scatterplot with aspect ratio one of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Football dataset.

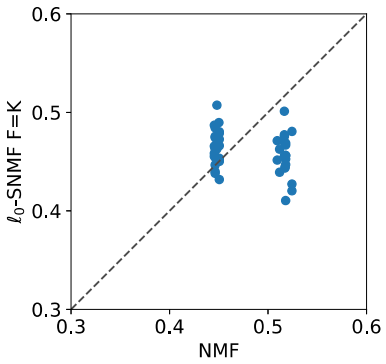


Figure 64: Scatterplot with aspect ratio one of NMI per repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the Polblogs dataset.

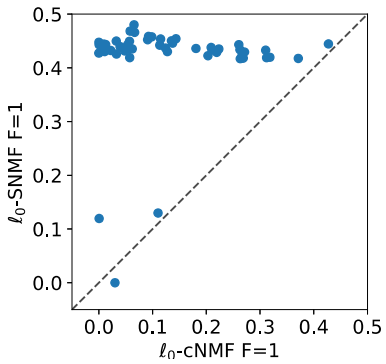


Figure 65: Scatterplot with aspect ratio one of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Polblogs dataset.

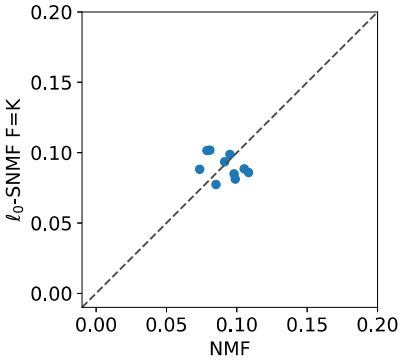


Figure 66: Scatterplot with aspect ratio one of NMI per repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the Cora dataset.

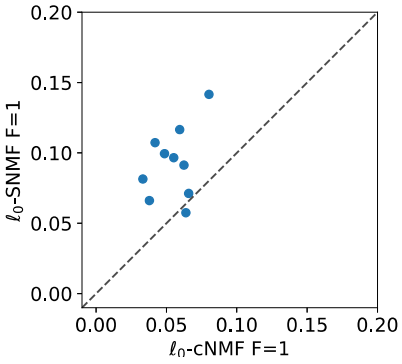


Figure 67: Scatterplot with aspect ratio one of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Cora dataset.

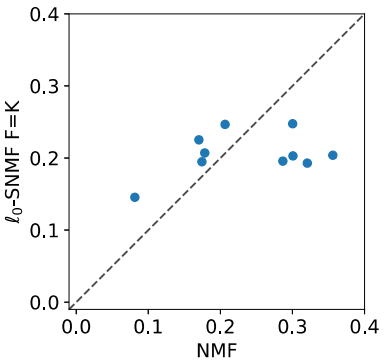


Figure 68: Scatterplot with aspect ratio one of NMI per repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the Citeseer dataset.

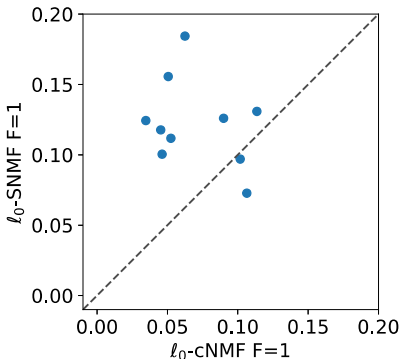


Figure 69: Scatterplot with aspect ratio one of NMI when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Citeseer dataset.

B.2.3 Normalized Loss

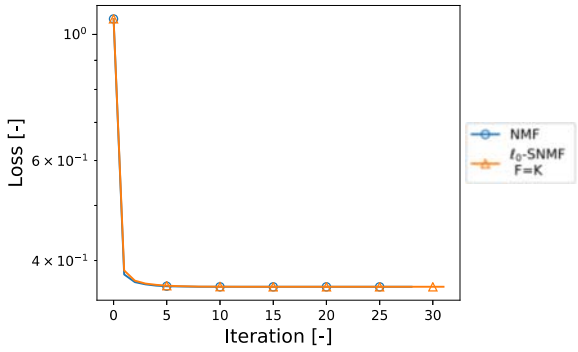


Figure 70: Normalized loss of one repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the Karate dataset.

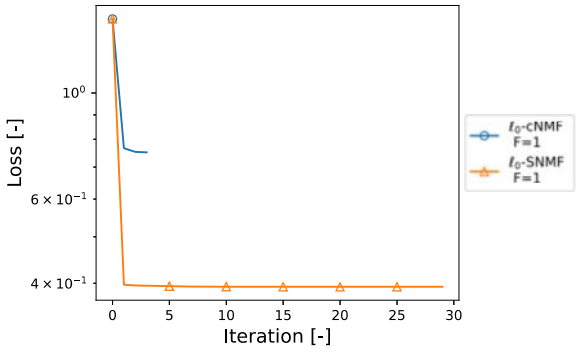


Figure 71: Normalized loss of one repetition when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Karate dataset.

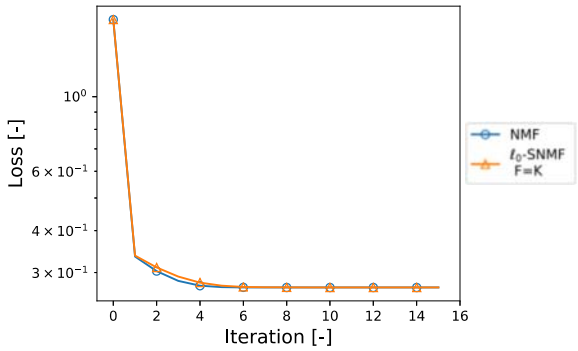


Figure 72: Normalized loss of one repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the Dolphins dataset.

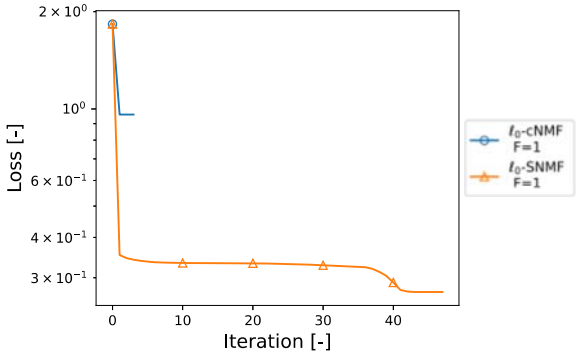


Figure 73: Normalized loss of one repetition when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Dolphins dataset.

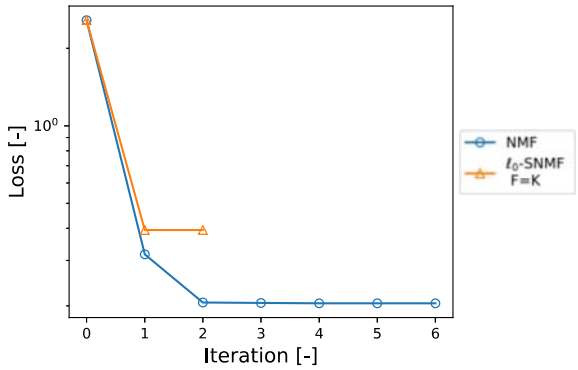


Figure 74: Normalized loss of one repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the Polbooks dataset.

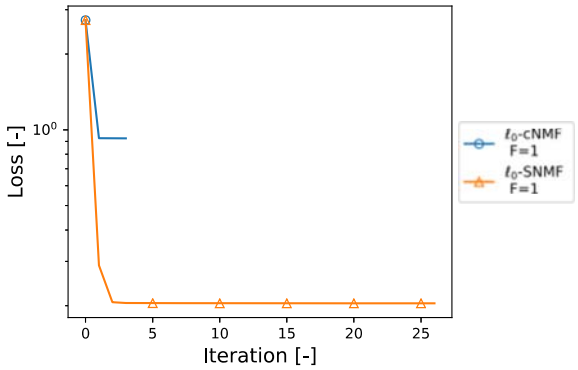


Figure 75: Normalized loss of one repetition when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Polbooks dataset.

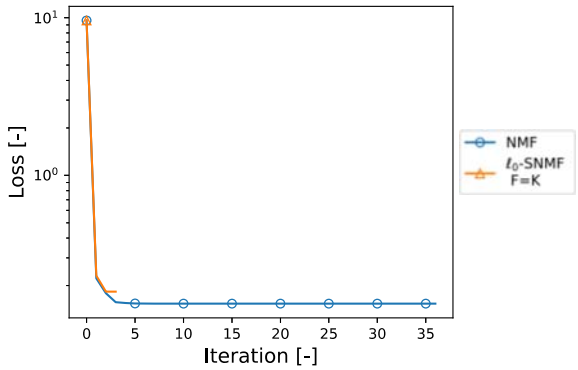


Figure 76: Normalized loss of one repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the Football dataset.

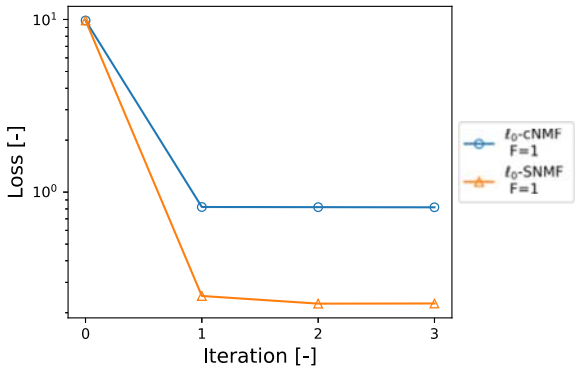


Figure 77: Normalized loss of one repetition when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Football dataset.

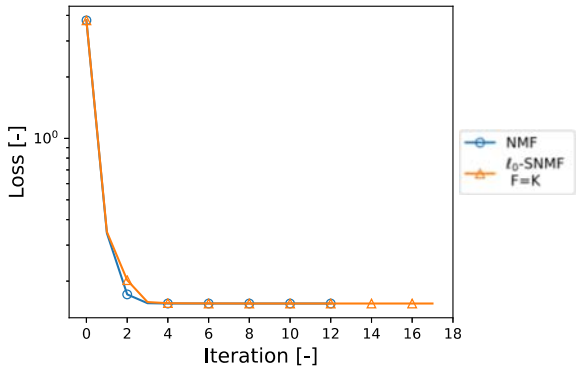


Figure 78: Normalized loss of one repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the Polblogs dataset.

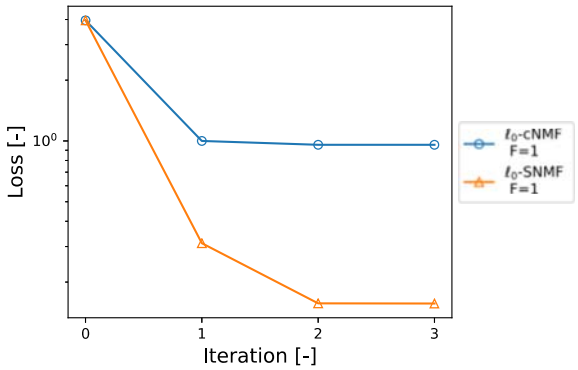


Figure 79: Normalized loss of one repetition when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Polblogs dataset.

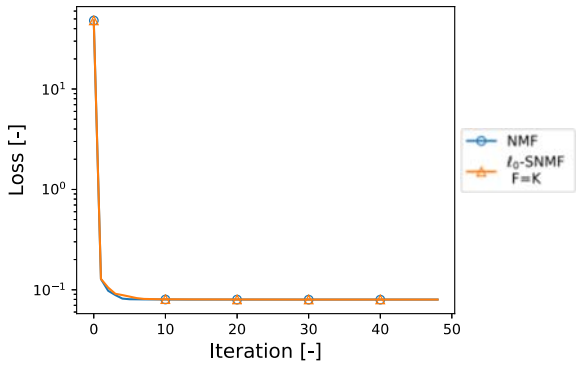


Figure 80: Normalized loss of one repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the Cora dataset.

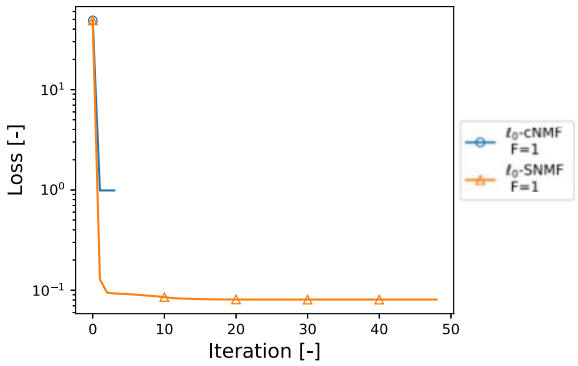


Figure 81: Normalized loss of one repetition when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Cora dataset.

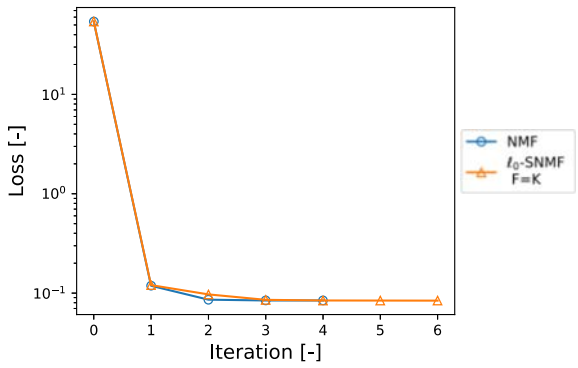


Figure 82: Normalized loss of one repetition when comparing NMF to ℓ_0-SNMF with $F = K$ for the Citeseer dataset.

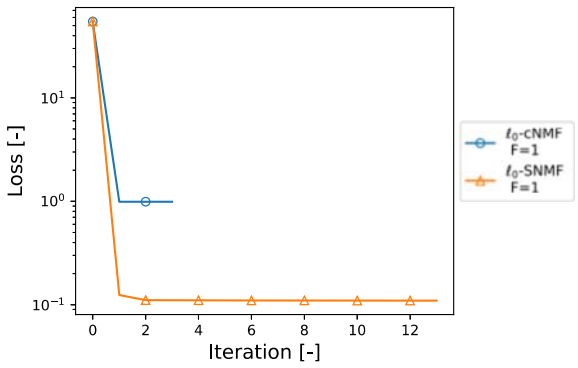


Figure 83: Normalized loss of one repetition when comparing ℓ_0-SNMF and ℓ_0-cNMF with $F = 1$ for the Citeseer dataset.