Nonlinear Feature Diffusion on Hypergraphs

Konstantin Prokopchik [1] **Austin R. Benson** [2] **Francesco Tudisco** [1]

Abstract

Hypergraphs are a common model for multiway relationships in data, and hypergraph semi-supervised learning is the problem of assigning labels to all nodes in a hypergraph, given labels on just a few nodes. Diffusions and label spreading are classical techniques for semi-supervised learning in the graph setting, and there are some standard ways to extend them to hypergraphs. However, these methods are linear models, and do not offer an obvious way of incorporating node features for making predictions. Here, we develop a nonlinear diffusion process on hypergraphs that spreads both features and labels following the hypergraph structure. Even though the process is nonlinear, we show global convergence to a unique limiting point for a broad class of nonlinearities and we show that such limit is the global minimum of a new regularized semi-supervised learning loss function which aims at reducing a generalized form of variance of the node features across the hyperedges. The limiting point serves as a node embedding from which we make predictions with a linear model. Our approach is competitive with popular graph and hypergraph neural network baselines, and also takes less time to train.

1. Introduction

In graph-based semi-supervised learning (SSL), one has labels on a small number of nodes, and the goal is to predict labels for the remaining nodes. Diffusions, label spreading, and label propagation are classical techniques for this problem, where known labels are diffused, spread, or propagated over the edges in a graph (Zhou et al., 2004; Zhu et al., 2003). These methods were originally developed for graphs where the set of nodes corresponds to a point cloud, and edges are similarity measures such as ϵ-nearest neighbors; however, they can also be used with relational data such as social networks or co-purchasing (Chin et al., 2019; Gleich & Mahoney, 2015; Juan et al., 2020; Kyng et al., 2015). In the latter case, diffusions are particularly well-suited because they directly capture the idea of homophily (McPherson et al., 2001) or assortativity (Newman, 2002), where labels are smooth over the graph.

While graphs are widely-used models for relational data, many complex systems and datasets are better described by higher-order relations that go beyond pairwise interactions (Battiston et al., 2020; Benson et al., 2018; Torres et al., 2021). For instance, co-authorship often involves more than two authors, people in social network gather in small groups and not just pairs, and emails can have several recipients. A hypergraph is a standard representation for such data, where a hyperedge can connect any number of nodes. Directly modeling higher-order interactions has led to improvements in several machine learning settings (Zhou et al., 2007; Benson et al., 2016; Li & Milenkovic, 2017; 2018; Yadati et al., 2019; Srinivasan et al., 2021).

Along this line, there are a number of diffusions or Laplacian-like regularization techniques for SSL on hypergraphs (Zhou et al., 2007; Hein et al., 2013; Zhang et al., 2017; Li et al., 2020; Liu et al., 2021; Veldt et al., 2020; Tudisco et al., 2021), which are also built on principles of similarity or assortativity. These methods are based on the optimization of a quadratic Laplacian-based regularized loss which enforces local and global consistency across the hypergredges. This optimization formulation makes it easy to interpret and to provide theoretical guarantees on the learning strategy. However, unlike the graph case, non-quadratic hypergraph consistency losses are typically required to model real-world data interactions at higher-order (Chan & Liang, 2020; Neuhäuser et al., 2022; Tudisco & Higham, 2021). Thus, the corresponding diffusion algorithms, based on e.g. gradient flow integration, are not linear and are in general no longer easy to interpret nor to analyze theoretically. Moreover, these methods are topically designed for cases where only labels are available, and do not take advantage of rich features or metadata associated with hypergraphs that are potentially useful for making ac-

[*]Equal contribution [1]Gran Sasso Science Institute, L'Aquila, Italy [2]Cornell University, New York, USA. Correspondence to: Konstantin Prokopchik <konstantin.prokopchik@gssi.it>, Austin R. Benson <arb@cs.cornell.edu >, Francesco Tudisco <francesco.tudisco@gssi.it>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

curate predictions. For instance, co-authorship or email data could have rich textual information.

Graph and hypergraph neural network (GNN) is a popular approach that uses both features and network structure for SSL (Yadati et al., 2019; Feng et al., 2019; Dong et al., 2020; Chien et al., 2022; Huang & Yang, 2021; Zhang et al., 2020a). Hidden layers of GNNs aggregate the features of neighboring nodes via neural networks and learn the model parameters by fitting to the labeled nodes. While combining features according to the hypergraph structure is a key idea, the loss function of typical GNNs does not take directly into account of the fact that connected nodes likely share similar labels. Moreover, GNNs can be expensive to train. In contrast, diffusion methods work precisely because of homophily and are typically fast. In the simple case of graphs, combining these two ideas has led to several recent advances (Klicpera et al., 2018; Huang et al., 2020; Jia & Benson, 2022).

Here, we combine the ideas of GNNs and diffusions for SSL on hypergraphs with a method that simultaneously diffuses both labels and features according to the hypergraph structure. In addition to incorporating features, our new diffusion can incorporate a broad class of nonlinearities to increase the modeling capability, which is critical to the architectures of both graph and hypergraph neural networks. The limiting point of the process provides an embedding at each node, which can then be combined with a simpler model such as multinomial logistic regression to make predictions at each node. This results into a method which is much faster than typical GNNs as the training phase and embedding computation are decoupled.

Remarkably, even though our model is nonlinear, we can still prove a number of theoretical properties about the diffusion process. In particular, we show that the limiting point of the process is unique and provide a simple, globally convergent iterative algorithm for computing it. Furthermore, we show that this limiting point is the global optimum of an interpretable hypergraph SSL loss function defined as the combination of a data fitting term and a Laplacian-like regularizer which aims at reducing a form of "generalized variance" on each hyperedge. Empirically, we find that using the limiting point of our nonlinear hypergraph diffusion as features for a linear model is competitive with a variety of graph and hypergraph neural network baselines and other diffusion algorithms on several real-world datasets. We also study the effect of the input feature embedding on the classification performance by either removing or modifying the node features. In particular, including the final embedding of hypergraph GNNs as additional features in the diffusion model does not improve accuracy, which provides evidence that our model is sufficient for empirical data.

2. Problem Set-up

We consider the multi-class semi-supervised classification problem on a hypergraph, in which we are given nodes with features and hyperedges connecting them. A small number of node labels are available and the goal is to assign labels to the remaining nodes.

Let $H = (V, E)$ be a hypergraph where $V = \{1, \dots, n\}$ is the set of nodes and $E = \{e_1, \dots, e_m\}$ is the set of hyperedges. Each hyperedge $e \in E$ has an associated positive weight $w(e) > 0$. In our setting every node can belong to an arbitrary number of hyperedges. Let δ_i denote the (hyper)degree of node $i \in V$, i.e., the weighted number of hyperedges node i participates in, $\delta_i = \sum_{e:i\in e} w(e)$, and let $D \in \mathbb{R}^{n\times n}$ be the diagonal matrix of the node degrees, i.e., $D = \mathrm{Diag}(\delta_1, \dots, \delta_n)$. Throughout we assume that hypergraph has no isolated nodes, i.e., $\delta_i > 0$ for all i. This is a standard assumption, as one can always add self-loops or remove isolated vertices. Let K denote the $n \times m$ incidence matrix of H, whose rows correspond to nodes and columns to hyperedges:

$$K_{i,e} = \begin{cases} 1 & i \in e \\ 0 & \text{otherwise.} \end{cases}$$

To include possible weights on hyperedges, we use a diagonal matrix W defined by $W = \mathrm{Diag}(w(e_1), \dots, w(e_m))$.

We will represent d-dimensional features on nodes in H by a matrix $X \in \mathbb{R}^{n\times d}$, where row $x_i = X_{i,:} \in \mathbb{R}^d$ is the feature vector of $i \in V$. Suppose each node belongs to one of c classes, denoted as $\{1, \dots, c\}$, and we know the label of a (small) training subset $\mathcal{T}$ of the nodes V. We denote by $Y \in \mathbb{R}^{n\times c}$ the input-labels matrix of the nodes, with rows y_i entrywise defined by $Y_{ij} = (y_i)_j = 1$ if node i belongs to class j, and $(y_i)_j = 0$ otherwise. Since we only know the labels for the nodes in $\mathcal{T}$, all the y_i for $i \notin \mathcal{T}$ are zero, while the for $i \in \mathcal{T}$, y_i has exactly one nonzero entry (one-hot encoding).

3. Background and Related Work

We review basic ideas in hypergraph neural networks (HNNs) for SSL and hypergraph label spreading (HLS), which will contextualize the method we develop next.

3.1. Neural Network Approaches

Graph neural networks are broadly adopted methods for semi-supervised learning on graphs. Several generalizations to hypergraphs have been proposed, and we summarize the most fundamental ideas here.

When $|e| = 2$ for all $e \in E$, the hypergraph is a standard graph G. The basic formulation of a graph convolutional network (GCN) (Kipf & Welling, 2017) is based on a first-

order approximation of the convolution operator on graph signals (Mallat, 1999). This approximation boils down to a mapping given by the normalized adjacency matrix of the graph $\overline{A} = I - L$, where L is the (possibly rescaled) normalized Laplacian $L = I - D^{-1/2}AD^{-1/2}$ and A is the adjacency matrix. The forward model for a two-layer GCN is then

$$Z = \operatorname{softmax}(F) = \operatorname{softmax}(\overline{A}\sigma(\overline{A}X\,\Theta^{(1)})\Theta^{(2)})$$

where $X \in \mathbb{R}^{n\times d}$ is the matrix of the node features, $\Theta^{(1)}$, $\Theta^{(2)}$ are the input-to-hidden and hidden-to-output weight matrices of the network and σ is a nonlinear activation function. Here, the approximated graph convolutional filter $\overline{A}$ combines features across nodes that are well connected in the graph. For multi-class semi-supervised learning problems, the weights $\Theta^{(i)}$ are then trained minimizing the cross-entropy loss over the training set of known labels $\mathcal{T}$.

Several hypergraph variations of this neural network model have been proposed for the more general case $|e| \geq 2$. A common strategy is to consider a hypergraph Laplacian L and define an analogous convolutional filter. One relatively simple approach is to define L as the Laplacian of the clique expanded graph of H (Agarwal et al., 2006; Zhou et al., 2007), where the hypergraph is mapped to a graph on the same set of nodes by adding a clique among the nodes of each hyperedge. This is the approach used in HGNN (Feng et al., 2019). Other variants use mediators or general permutation-invariant aggregating functions instead of cliques in the hypergraph to perform the graph reduction (Chan & Liang, 2020; Huang et al., 2020), or learn the hypergraph convolutional filter via a suitable attention-based multi-set function architecture (Chien et al., 2022). HyperGCN (Yadati et al., 2019) is based on the nonlinear hypergraph Laplacian proposed in (Chan et al., 2018; Louis, 2015). This model uses a GCN on a reduced graph $G_X = (V, E_X)$ that depends on the features, where $(u, v) \in E_X$ if and only if $(u, v) = \arg\max_{i,j\in e} \|x_i - x_j\|$, for all hyperedges e of the original hypergraph. An approximate graph convolutional filter $\overline{\mathcal{A}}_X$ is then defined in terms of the normalized Laplacian of G_X as before. Thus, the two-layer forward model for HyperGCN is

$$Z = \operatorname{softmax}(\mathcal{A}_{F^{(1)}}\Theta^{(2)}), \quad F^{(1)} = \sigma(\mathcal{A}_X\Theta^{(1)})\,.$$

3.2. Laplacian Regularization and Label Spreading

Semi-supervised learning based on Laplacian-like regularization strategies were developed in (Zhou et al., 2004) for graphs and then in (Zhou et al., 2007) for hypergraphs. The main idea of these approaches is to obtain a classifier F by minimizing the regularized square loss function

$$\min_F \ell_\Omega := \|F - Y\|^2 + \lambda\,\Omega(F) \tag{1}$$

where Ω is a regularization term that takes into account for the hypergraph structure. (Note that only labels — and not features — are used here.) In particular, if $f_i = F_{i,:}$ denotes the i-th row of F, the clique expansion approach of (Zhou et al., 2007) defines $\Omega = \Omega_{L^2}$, with

$$\Omega_{L^2}(F) = \textstyle\sum_{e\in E}\sum_{i,j\in e}\frac{w(e)}{|e|}\left\|\frac{f_i}{\sqrt{\delta_i}} - \frac{f_j}{\sqrt{\delta_j}}\right\|^2,$$

while the total variation on hypergraph regularizer proposed in (Hein et al., 2013) is $\Omega = \Omega_{TV}$, where

$$\Omega_{TV}(F) = \textstyle\sum_{e\in E} w(e)\max_{i,j\in e}\|f_i - f_j\|^2$$

The function Ω_{L^2} is quadratic, so its gradient is a rescaled version of the Laplacian of the clique expanded graph of H. Thus, HGNN is implicitly applying this regularization. Similarly, the graph construction in HyperGCN is implicitly applying a regularization based on the hypergraph total variation Ω_{TV}.

These two choices of regularizing terms can be solved by means of different strategies. As Ω_{L^2} is quadratic, one can solve (1) via gradient descent with the learning rate $\alpha = \lambda/(1+\lambda)$ to obtain the simple method

$$F^{(k+1)} = \alpha\overline{A}_H F^{(k)} + (1-\alpha)Y, \tag{2}$$

where $\overline{A}_H$ is the normalized adjacency matrix of the clique-expanded graph of H. The sequence (2) converges to the global solution $F_\star$ of (1) for any starting point and the limit $F_\star$ is entrywise nonnegative. This method is usually referred to as Hypergraph Label Spreading (HLS) as the iteration in (2) takes the initial labels Y and "spreads" or "diffuses" them throughout the vertices of the hypergraph H, following the edge structure of its clique-expanded graph.

The total variation inspired regularizer Ω_{TV} is related to the 1-Laplacian energy (Bühler & Hein, 2009; Tudisco & Hein, 2018) and has advantages for hyperedge cut interpretations. However, although Ω_{TV} is convex, it is not differentiable, and computing (1) requires more complex and computationally demanding methods (Hein et al., 2013; Zhang et al., 2017). Unlike HLS in (2), this is not easily interpreted as a label diffusion.

4. Hyperedge Variance Regularization and Nonlinear Diffusion

In this section, we propose a new hypergraph regularization term Ω_μ which, rather than minimizing the distance between each node pair on a hyperedge, aims at reducing the variance (or a generalized variance) across the hyperedge nodes. Precisely, consider the regularization term of the form

$$\Omega_\mu(F) = \sum_{e\in E}\sum_{i\in e} w(e)\left\|\frac{f_i}{\sqrt{\delta_i}} - \mu\Big(\Big\{\frac{f_j}{\sqrt{\delta_j}} : j\in e\Big\}\Big)\right\|^2 \tag{3}$$

where $\mu(\cdot)$ is a function that combines node embeddings on each hyperedge. When μ is the mean $\mu(\{z_j : j \in e\}) = \frac{1}{|e|}\sum_{j\in e} z_j$, the right hand side in (3) coincides exactly with the variance of $f_i/\sqrt{\delta_i}$ on the hyperedge e.

Note that, if F is the matrix with rows $F_{i,:} = f_i$, the mean of $f_i/\sqrt{\delta_i}$ over the hyperedge e can be written as the e-th row of the matrix $D_E^{-1}K^\top D^{-1/2}F$, where D_E denotes the $m \times m$ diagonal matrix with diagonal entries $D_{e,e} = |e|$. Here we use this observation to define a family of functions μ that generalizes the mean. Precisely, let

$$\mu_{\sigma,\varrho}\Big(\Big\{\frac{f_j}{\sqrt{\delta_j}} : j \in e\Big\}\Big) = \sigma(K^\top \varrho(D^{-1/2}F))_{e,:} \qquad (4)$$

where σ and ϱ are (in general, nonlinear) operators that describe the way an embedding F is transformed and combined across the hyperedges. For example, when σ and ϱ are diagonal operators (similar to activation functions) — i.e. they are such that $\sigma(F)_{ij} = \sigma_i(F_{ij})$ for some functions $\sigma_1, \sigma_2, \cdots : \mathbb{R} \to \mathbb{R}$ (and the same for ϱ) — we have that

$$\mu_{\sigma,\varrho}\Big(\Big\{\frac{f_j}{\sqrt{\delta_j}} : j \in e\Big\}\Big) = \sigma_e\Big(\sum_{j\in e} \varrho_j\big(\frac{f_j}{\sqrt{\delta_i}}\big)\Big).$$

An important example of σ and ϱ of this form, which we will use in all our experiments, is

$$\varrho(Z_1) := Z_1^p, \qquad \sigma(Z_2) := (D_E^{-1}Z_2)^{1/p}, \qquad (5)$$

where the powers are taken entry-wise and Z_i are matrices of appropriate size. With these choices, for every $e \in E$ we have that $\mu_{\sigma,\varrho}(\{f_i/\sqrt{\delta_i}, i \in e\}) = \mathrm{mean}_p\{f_i/\sqrt{\delta_i} : i \in e\}$ is the p-power mean of the normalized feature vectors $f_i/\sqrt{\delta_i}$ of the nodes i in the hyperedge e, where

$$\mathrm{mean}_p\{z_i : i \in e\} = \big(\tfrac{1}{|e|}\textstyle\sum_{i\in e} z_i^p\big)^{1/p}.$$

With this choice of σ and ϱ the regularization term (3) reads

$$\Omega_{\mu_{\sigma,\varrho}}(F) = \sum_{e\in E}\sum_{i\in e} w(e)\Big\|\frac{f_i}{\sqrt{\delta_i}} - \mathrm{mean}_p\Big\{\frac{f_j}{\sqrt{\delta_j}} : j \in e\Big\}\Big\|^2 \qquad (6)$$

that is, the embedding F that minimizes $\Omega_{\mu_{\sigma,\varrho}}$ minimizes the variation of each node embedding f_i from the p-power mean of the embeddings of the nodes in each hyperedge i participates to. In particular, when $p = 2$, the regularization term (6) computes the variance of all the nodes in each of the hyperedges. Note that, for nonnegative embeddings, other special cases of mean_p include the geometric mean (for $p \to 0$), the harmonic mean (for $p = -1$) as well as the minimum and maximum functions (for $p \to -\infty$ and $p \to +\infty$, respectively).

4.1. Nonlinear Diffusion Method

Consider the regularized loss function ℓ_Ω in (1) with $\Omega = \Omega_{\mu_{\sigma,\varrho}}$. Unlike the clique-expansion case, $\Omega = \Omega_{L^2}$, $\ell_{\Omega_{\mu_{\sigma,\varrho}}}$ is non-quadratic and non-convex in general. Despite this fact, we will show below that the global solution to $\min_F \ell_{\Omega_{\mu_{\sigma,\varrho}}}(F)$ can be computed via a simple hypergraph diffusion algorithm similar to a nonlinear version of HLS (2), provided we restrict to the set of embeddings with nonnegative entries. We introduce the diffusion model next.

Recall that in our setting each node $i \in V$ has a label-encoding vector y_i (y_i is the all-zero vector for the initially unlabeled points $i \notin \mathcal{T}$) and a feature vector x_i. Thus, each node in the hypergraph has an initial $(c+d)$-dimensional embedding, which forms an input matrix $U = [Y \;\; X]$.

The proposed hypergraph semi-supervised classifier uses the normalized limit point of the nonlinear diffusion process

$$F^{(k+1)} = \alpha\,\mathcal{L}(F^{(k)}) + (1-\alpha)\,U\,. \qquad (7)$$

where the nonlinear diffusion map $\mathcal{L}$ is a form of nonlinear Laplacian operator defined as

$$\mathcal{L}(F) = D^{-1/2}KW\sigma(K^\top\varrho(D^{-1/2}F))\,. \qquad (8)$$

The limit point of the diffusion process (7) results in a new embedding $F_\star = [Y_\star\, X_\star] \in \mathbb{R}^{n\times(c+d)}$. We will show in §4.3 that such limit exists, is unique and minimizes a normalized version of the SSL regularized loss $\ell_{\Omega_{\mu_{\sigma,\varrho}}}$. We will then use $F_\star$ to train a logistic multi-class classifier based on the known labels $i \in \mathcal{T}$ and their new embedding $F_\star$. Thus, unlike GNNs, the training phase and the computation of the embedding $F_\star$ are decoupled, and thus are much faster. The overall SSL algorithm is detailed in Algorithm 1. Note that the convergence of (7) is not trivial, due to the nonlinearity of $\mathcal{L}$. We further comment on this issue in the appendix.

4.2. Related Nonlinear Diffusion Models

Our nonlinear diffusion process (7) propagates both input node label and feature embeddings through the hypergraph in a manner similar to (2), but allowing for nonlinear activations, which increases modeling power. Firstly, $\mathcal{L}$ is a generalization of the normalized adjacency matrix of the clique-expansion hypergraph $\overline{A}_H$ (2). Secondly, for a standard graph, i.e., a hypergraph where all the edges have exactly two nodes, $KWK^\top = A + D$ where A is the adjacency matrix of the graph and D is the diagonal matrix of the weighted node degrees. Similarly, for a general hypergraph H, we have the identity $KWK^\top = A_H + D$, where A_H is the adjacency matrix of the clique-expanded graph associated with H. Then, when $\sigma = \varrho = \mathrm{id}$, $\mathcal{L}$ coincides with

$$D^{-1/2}KWK^\top D^{-1/2} = \overline{A}_H + I\,, \qquad (9)$$

Algorithm 1 HyperND: Nonlinear Hypergraph Diffusion

Input:
- Incidence matrix K and weights matrix W;
- mixing mappings σ, ϱ as in (5);
- normalization mapping φ as in (10);
- label $Y \in \{0,1\}^{n\times c}$ and feature $X \in \mathbb{R}^{n\times d}$ matrices;
- regularization coefficient $\alpha \in (0,1)$ and stopping tolerance tol.

Shift and scale U to obtain $U > 0$, e.g., via (12) when $X \geq 0$
$U \leftarrow U/\varphi(U)$
$F^{(0)} \leftarrow U$
repeat
  $G \leftarrow \alpha\mathcal{L}(F^{(k)}) + (1-\alpha)U$
  $F^{(k+1)} \leftarrow G/\varphi(G)$
until $\|F^{(k+1)} - F^{(k)}\|/\|F^{(k+1)}\| < \text{tol}$
New node embedding: $F_\star = [Y_\star\ X_\star] \leftarrow F^{(k+1)}$
Optimize Θ for $Z_\star \leftarrow \text{softmax}(F_\star\Theta)$ to minimize cross-entropy.

Output:
Prediction $\arg\max_c(Z_\star)_{i,c}$ for class of unlabeled node i.

the normalized adjacency matrix of the clique-expansion hypergraph (Zhou et al., 2007; Feng et al., 2019) as in (2).

When σ and ϱ are not linear, $\mathcal{L}$ can represent a broad family of nonlinear diffusion mappings, with special cases used in different contexts. For example, in the graph case, if $\varrho = \text{id}$ and $\sigma(x) = |x|^{p-1}\text{sign}(x)$, then $\mathcal{L}$ boils down to the graph p-Laplacian operator (Saito et al., 2018; Elmoataz et al., 2008; Bühler & Hein, 2009). Exponential and logarithmic based choices of σ and ϱ give rise to nonlinear Laplacians used to model chemical reactions (Rao et al., 2013; Van der Schaft et al., 2016) as well as to model consensus dynamics and opinion formation in hypergraphs (Neuhäuser et al., 2022). Trigonometric functions such as $\sigma(x) = \sin(x)$ are used to model graph and hypergraph oscillators (Schaub et al., 2016; Battiston et al., 2021; Millán et al., 2020). In the context of semi-supervised learning, nonlinear diffusion mappings based on entrywise powers and generalized means are used, for example in (Ibrahim & Gleich, 2019; Tudisco et al., 2021). Similarly to our proposed operator, the diffusion map of (Tudisco et al., 2021) generalizes the classical linear label spreading to 3-uniform hypergraphs by considering an adjacency tensor-based model that combines labels on hyperedges via p-power means. We notice that our proposed diffusion operator (8) extends the one proposed by Tudisco et al. (2021). In fact, when restricted to uniform hypergraphs and for $\varrho = I$, the two mappings coincide, modulo relatively minor adjustments. However, extending the model there proposed to the case of hyperedges of arbitrary size would require one to split the hyperedges into batches of same sizes and compute the corresponding adjacency tensors. Compared to our proposed incidence matrix model, this is computationally significantly more demanding both because it requires the computation of several high order tensors and because the multiplication operations with tensors are more expensive than those with matrices (which use fast BLAS routines from, e.g., NumPy), especially when used to propagate high-dimensional features rather than only labels.

4.3. Convergence

The convergence of (7) is discussed in Theorem 4.1 below, where we show that, under mild assumptions on σ and ϱ, (7) always converges, provided the embedding is suitably normalized at each step. Moreover, the result shows that the nonlinear hypergraph filter $\mathcal{L}$ eventually generates an embedding that minimizes a regularized loss function of the form (1), with regularization term $\Omega = \Omega_{\mu_{\sigma,\varrho}}$. The proof of Theorem 4.1 is based on Thm. 3.1 by Tudisco et al. (2021) and is moved to the appendix.

In the following, we write $F \geq 0$ (resp. $F > 0$) to indicate that F has nonnegative (resp. positive) entries. Moreover, we write $\sigma \in \hom_+(a)$ to denote that σ is positive and homogeneous of degree a, i.e. that the following holds for σ: (1) $\sigma(F) > 0$ for all $F > 0$; and (2) $\sigma(\lambda F) = \lambda^a\sigma(F)$ for all $\lambda > 0$ and all $F > 0$.

Note that the class of operators $\hom_+(a)$ is quite general and it includes, for example different forms of LeakyReLU functions $\sigma(F) = \max\{0, F^a\} \pm \varepsilon\max\{0, -F^a\}$ as well as the family of homogeneous nonnegative generalized polynomial (polynomial with real powers) operators, defined as

$$\sigma(F)_{i,:} = \sum_{j=1}^{n} B^{(i,j)} f_1^{\alpha_1^{(i,j)}} \cdots f_n^{\alpha_n^{(i,j)}}$$

for any nonnegative coefficients $\alpha_j^{(i,j)}$ and any nonnegative matrices $B^{(i,j)}$, as long as $\sum \alpha_1^{(i,j)} + \cdots + \alpha_n^{(i,j)} = a$, i.e. the sum of the powers is constant for all i, j, to ensure $\sigma(\lambda F) = \lambda^a\sigma(F)$ for all $\lambda > 0$.

Theorem 4.1. *Let $\mathcal{L}$ be defined as in (8) and let $\mu_{\sigma,\varrho}$ be defined as in (4). Let $f_i = F_{i,:}$ denote the i-th row of F and define the real-valued function*

$$\varphi(F) = 2\sqrt{\sum_{e\in E} w(e)\left\|\mu_{\sigma,\varrho}\left(\left\{\frac{f_j}{\sqrt{\delta_j}}, j \in e\right\}\right)\right\|^2}. \quad (10)$$

Assume that $\sigma \in \hom_+(a)$ and $\rho \in \hom_+(b)$ for some $a, b \in \mathbb{R}$. Let U be an entrywise positive input embedding and let $\alpha \in [0,1]$. If $ab = 1$ and if $\mathcal{L}$ is differentiable and such that $\mathcal{L}(F) \geq \mathcal{L}(\widetilde{F})$ for all $F \geq \widetilde{F} > 0$, then for any starting point $F^{(0)} \geq 0$, the sequence

$$\begin{cases} \widetilde{F}^{(k)} = \alpha\,\mathcal{L}(F^{(k)}) + (1-\alpha)\,U \\ F^{(k+1)} = \widetilde{F}^{(k)}/\varphi(\widetilde{F}^{(k)}) \end{cases} \quad (11)$$

Table 1: Details for the real-world hypergraph datasets used in the experiments.

		DBLP co-authorship	**Pubmed** co-citation	**Cora** co-authorship	**Cora** co-citation	**Citeseer** co-citation	**Foodweb** carbon-exchange
$\lvert V\rvert$	(#nodes)	43413	19717	2708	2708	3312	122
$\lvert E\rvert$	(#hyperedges)	22535	7963	1072	1579	1079	141233
d	(#features)	1425	500	1433	1433	3703	0
c	(#labels)	6	3	7	7	6	3

converges to a unique fixed point $F_\star$ of (7), such that $\varphi(F_\star) = 1$, $F_\star > 0$. Moreover, $F_\star$ is the solution of

$$\begin{cases} \min\limits_{F} \ \left\| F - \dfrac{U}{\varphi(U)} \right\|^2 + \lambda\, \Omega_{\mu_{\sigma,\varrho}}(F) \\ \text{subject to } \ F \geq 0, \ \varphi(F) = 1, \end{cases}$$

where $\lambda = \alpha/(1-\alpha)$.

Note that the choices of σ and ϱ in (5), which lead to the p-power mean regularization term (6), and the corresponding diffusion operator $\mathcal{L}$ satisfy all the assumptions of Thm 4.1.

4.4. Algorithm Details and Limitations

Once the new node embedding $F_\star$ is computed via (11), we use it to infer the labels of the non-labeled data points via a simple softmax output layer which minimizes cross-entropy (see Algorithm 1).

Similar to HLS, the parameter α in Alg. 1 yields a convex combination of the diffusion mapping $\mathcal{L}$ and the "bias" U, allowing to tune the contribution given by the homophily along the hyperedges and the one provided by the input features and labels. In other words, in view of Theorem 4.1, α quantifies the strength of the regularization parameter λ, which allows us to tune the contribution of the regularization term $\Omega_{\mu_{\sigma,\varrho}}$ over the data-fitting term $\|F - U/\varphi(U)\|$.

A requirement for our main theoretical results is entrywise positivity of the input embedding U. While this is a limitation of the theory and the methods, it turns out to not be that stringent in practice. If $X \geq 0$, i.e. we have nonnegative node features, we can easily obtain a positive embedding by performing an initial label smoothing (Müller et al., 2019; Szegedy et al., 2016), i.e. we choose a small $\varepsilon > 0$ and set

$$U_\varepsilon = (1-\varepsilon)[Y \ X] + \varepsilon \mathbb{1}\mathbb{1}\,, \tag{12}$$

being $\mathbb{1}\mathbb{1}$ the all-one matrix of the appropriate size. Note that nonnegative input features $X \geq 0$ are not uncommon. For instance, bag-of-words, one-hot encodings, and binary features in general are all nonnegative. In fact, for all of the real-world datasets we consider in our experiments, the features are nonnegative. Similarly, if some of the input features have negative values (e.g., features coming from a word embedding), one could perform other preprocessing (e.g., shift on all embeddings) to get the required $[Y\,X] > 0$.

Another implicit requirement of the proposed method is the assumption that the hypergraph datasets at hand are homophilic. This is a limitation shared by many graph and hypergraph learning algorithms, although relevant heterophilic settings exist in the real-world, see e.g. (Zhu et al., 2020).

5. Experiments

We now evaluate our method on several real-world hypergraph datasets (Table 1). We use five co-citation and co-authorship hypergraphs: Cora co-authorship, Cora co-citation, Citeseer, Pubmed (Sen et al., 2008) and DBLP (Rossi & Ahmed, 2015). All nodes in the datasets are documents, features are given by the content of the abstract and hyperedge connections are based on either co-citation or co-authorship. The task for each dataset is to predict the topic to which a document belongs. We also consider a foodweb hypergraph, where the nodes are organisms and hyperedges represent directed carbon exchange in the Florida bay (foo). Here we predict the role of the nodes in the food chain. This hypergraph has no features, so only labels are used for HyperND while we use a one-hot encoding for the baselines. The code implementing the experiments is available at `https://github.com/compile-gssi-lab/HyperND`.

We compare our method to six baselines. Two are hypergraph convolutional network, designed to work specifically for hypergraphs:

- **HGNN** (Feng et al., 2019) This is a hypergraph neural network model that uses the clique-expansion Laplacian (Zhou et al., 2007; Agarwal et al., 2006) for the hypergraph convolutional filter.
- **HyperGCN** (Yadati et al., 2019) This is a hypergraph convolutional network model with regularization similar to the total variation (see also §3). There are three architectural variants (1-HyperGCN, FastHyperGCN, HyperGCN), and we report whichever gives the best performance.

One is a hypergraph Laplacian regularization method:

- **HTV** (Zhang et al., 2017) This is a confidence-interval subgradient-based method that minimizes the 1-Laplacian

Table 2: Accuracy (mean ± standard deviation) over five random samples of the training nodes $\mathcal{T}$. We compare HyperND and the six baseline methods (APPNP, HGNN, HyperGCN, SGC, SCE, HTV). Overall, HyperND is more accurate than the baselines.

	Method	**HyperND**	**APPNP**	**HGNN**	**HyperGCN**	**SGC**	**SCE**	**HTV**
Data	% labeled							
Citeseer	4.2%	**72.13** ±1.00	63.51 ±1.39	61.78 ±3.46	50.94 ±8.27	52.66 ±2.18	61.28 ±1.61	29.63±0.3
Cora-author	5.2%	**77.33** ±1.51	71.34 ±1.60	63.11 ±2.73	61.27 ±1.06	30.46 ±0.22	71.96 ±2.18	44.55±0.6
Cora-cit	5.2%	**83.13** ±1.11	82.08 ±1.61	62.88 ±2.26	62.78 ±2.73	29.08 ±0.25	79.85 ±1.91	35.60±0.8
DBLP	4.0%	**89.63** ±0.12	88.94 ±0.07	73.82 ±0.71	70.02 ±0.10	43.61 ±0.17	87.50 ±0.19	45.19±0.9
Foodweb	5.0%	64.09 ±5.94	**69.12** ±3.30	57.09 ±2.33	56.14 ±3.85	57.45 ±0.47	63.50 ±4.78	57.23±0.9
Pubmed	0.8%	**82.81** ±2.16	81.50 ±1.18	72.57 ±1.03	78.11 ±0.99	54.30 ±1.11	77.57 ±2.34	47.04±0.8

Figure 1: Performance of the proposed HyperND for varying p and α parameters.

inspired loss (1) with $\Omega = \Omega_{TV}$. In our experiments, this method outperformed other label spreading and Laplacian regularization techniques such as the original PDHG strategy of Hein et al. (2013), the HLS method (Zhou et al., 2007) and the p-Laplacian approach of Saito et al. (2018).

The remaining three baselines, instead, are graph convolutional networks designed for graph data, which we apply to the clique-expanded version of the considered hypergraphs:

- **APPNP** (Klicpera et al., 2018) This is a graph convolutional network model combined with PageRank. The authors of this paper introduce a personalized propagation of neural predictions and its approximation based on the relationship between GCN and PageRank.
- **SGC** (Wu et al., 2019) This is a graph convolutional network model without nonlinearities.
- **SCE** (Zhang et al., 2020b) This is a graph convolutional network model inspired by a sparsest-cut problem, where unsupervised network embedding is learned only using negative samples for training.

Table 2 shows the size of the training dataset for each network and compares the accuracy (mean ± standard deviation) of HyperND against the different baselines. Precision and recall are reported in Tables 3 and 4 in the appendix. For each dataset, we use five trials with different samples of the training nodes $\mathcal{T}$. All of the algorithms that we use have hyperparameters. For the baselines we use the reported tuned hyperparameters. For all of the neural network-based models, we use two layers and 200 training epochs, following (Yadati et al., 2019) and (Feng et al., 2019). For our method and HTV, which have no training phase, we run 5-fold cross validation with label-balanced 50/50 splits to choose α from $\{0.1, 0.2, \ldots, 0.9\}$ and p from $\{1, 2, 3, 5, 10\}$. Precisely, we split the data into labeled and unlabeled points. We split the labeled points into training and validation sets of equal size (label-balanced 50/50 splits) and we choose the parameters based on the average performance on the validation set over 5 random repeats. Then, we assess the performance on the held out test set, which is comprised of all the initially non-labeled points. We repeat this process 5 times and we choose the value of α, p that gives the best mean accuracy. These values differ across the different random samplings of the training dataset. Thus, in order to highlight how the performance is affected by different values of α and p, we show in Figure 1 the mean accuracy over 10 runs of HyperND, for all of the considered values of α and p.

All the datasets we use here have nonnegative input embedding $[Y\ X]$ which we preprocess via label smoothing as in (12), with $\varepsilon = \texttt{1e} - \texttt{6}$. Our experiments have shown that different choices of ε do not influence the classification performance of the algorithm.

Due to its simple regularization interpretation, we choose σ and ϱ to be the p-power mean considered in (5), for various p. When varying p, we change the nonlinear activation functions that define the final embedding $F_\star$. Our proposed nonlinear diffusion method performs overall very well. Interestingly, the best competitors are not hypergraph-oriented methods but rather graph methods directly applied to the clique-expanded graph. In particular, APPNP is the strongest baseline, and this method also strongly relies on diffusions. Moreover, the poorer performance observed for

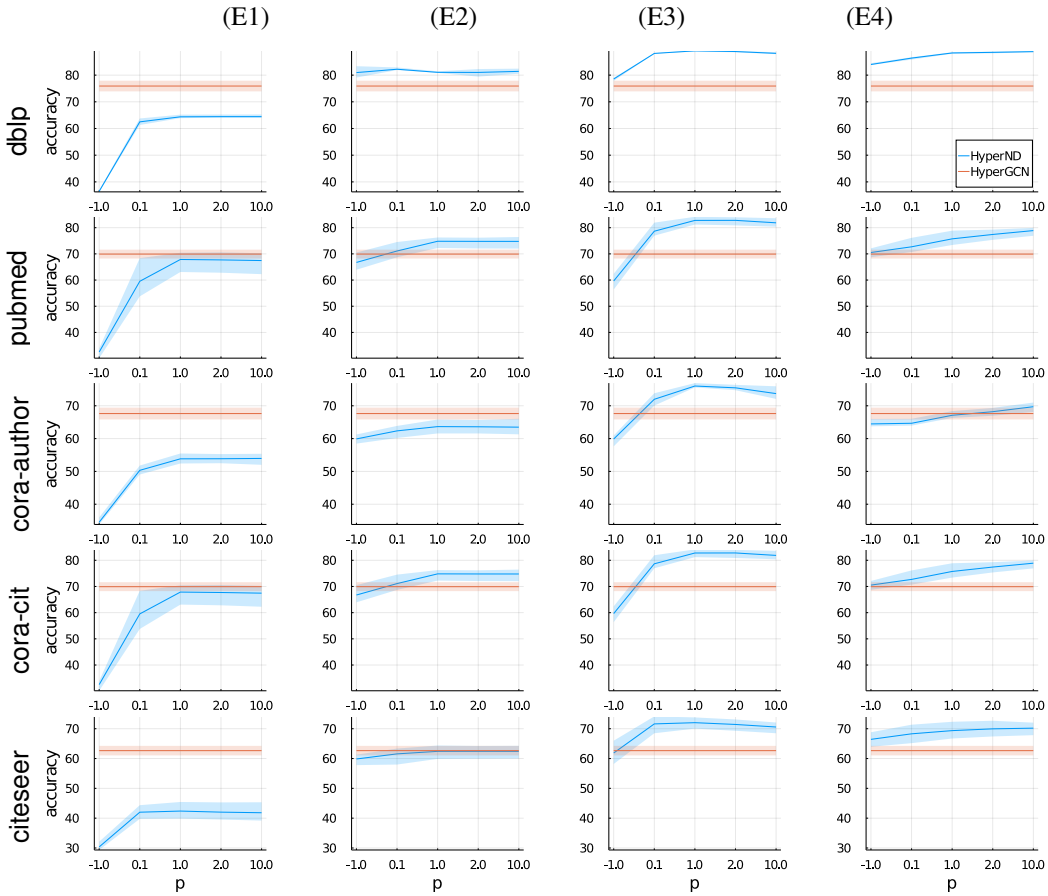

Figure 2: Accuracy (mean and standard deviation) of multinomial logistic regression classifier, using different combinations of features obtained from embeddings (E1)–(E4).

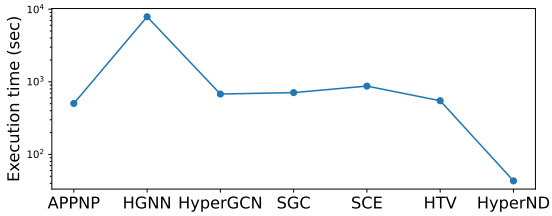

Figure 3: Execution time on the largest dataset DBLP (for one hyper-parameter setting in each case). All methods are comparable on small datasets.

food web dataset (which has no features) highlights the ability of HyperND to create meaningful feature embeddings, in addition to propagating labels.

We also point out that since HyperND is a forward model, it can be implemented efficiently. The cost of each iteration of (2) is dominated by the cost of the two matrix-vector products with the matrices K and $K^\top$, both of which only require a single pass over the input data and can be parallelized with standard techniques. Therefore, HyperND scales linearly with the number and size of the hyperedges, i.e. the size of the data. Thus, it is typically cheap to compute (similar to standard hypergraph LS) and is overall faster to train than a two-layer GCN. Training times are reported in Figure 3, where we compare mean execution time over ten runs for all the methods on DBLP. The execution times are similar on other datasets. For HyperND, we show mean execution time over the five random choices of p. HyperND is up to 2 order of magnitudes faster than the baselines.

To further highlight the learning capabilities of the diffusion map we are proposing, we present one more test below. The diffusion map (2) generates a new embedding $F_\star = [Y_\star \; X_\star]$ from the fixed point of a purely forward model. This model yields a new feature-based representation $X_\star$, similar to the last-layer embedding of any neural network approach. A natural question is whether or not $X_\star$ is actually a better embedding. To this end, in the next experiment we consider four node embeddings $\overline{F}$ and train a classifier via cross-entropy minimization of $Z = \text{softmax}(\overline{F}\Theta)$, optimizing Θ. Specifically, we consider the following:

(E1) $\overline{F} = Y_\star$. We run a nonlinear "purely label" spreading iteration, by setting $U = Y$ in (7).By Theorem 4.1, this embedding is a Laplacian regularization method.

(E2) $\overline{F} = [Y_\star\ X_{\text{hgcn}}]$, where X_{hgcn} is the embedding generated by HyperGCN before the softmax.

(E3) $\overline{F} = F_\star = [Y_\star\ X_\star]$, the limit point (7) of our HyperND. This is the embedding used for the results in Table 2 and Figure 1.

(E4) $\overline{F} = [Y_\star\ X_\star\ X_{\text{hgcn}}]$. This combines the representations of our HyperND and HyperGCN.

Figure 2 shows the accuracy for these embeddings with various values of p for the p-power mean in HyperND. The best performance is obtained by the two embeddings that contain our learned features $X_\star$: (E3) and (E4). In particular, while (E4) includes the final-layer embedding of HyperGCN, it does not improve accuracy over (E3).

6. Conclusions

Graph neural networks and hypergraph label spreading are two distinct techniques with different advantages for semi-supervised learning with higher-order relational data. The proposed diffusion approach (HyperND) tries to combine advantages from both approaches: feature-based learning, modeling flexibility, label-based regularization, and computational speed. Importantly, we can prove that the diffusion converges to an embedding that is the global minimizer of an interpretable regularized loss function which enforces small variance across the hyperedges, and we have an algorithm that can compute this optimal embedding. Overall, HyperND outperforms a variety of baseline neural network and label spreading based methods on several datasets.

References

Florida bay trophic exchange matrix. http://vlado.fmf.uni-lj.si/pub/networks/data/bio/foodweb/Florida.paj.

Agarwal, S., Branson, K., and Belongie, S. Higher order learning with graphs. In *Proceedings of the 23rd International Conference on Machine Learning*, pp. 17–24, 2006.

Battiston, F., Cencetti, G., Iacopini, I., Latora, V., Lucas, M., Patania, A., Young, J.-G., and Petri, G. Networks beyond pairwise interactions: Structure and dynamics. *Physics Reports*, 874:1–92, 2020.

Battiston, F., Amico, E., Barrat, A., Bianconi, G., Ferraz de Arruda, G., Franceschiello, B., Iacopini, I., Kéfi, S., Latora, V., Moreno, Y., et al. The physics of higher-order interactions in complex systems. *Nature Physics*, 17(10): 1093–1098, 2021.

Benson, A. R., Gleich, D. F., and Leskovec, J. Higher-order organization of complex networks. *Science*, 353(6295): 163–166, 2016.

Benson, A. R., Abebe, R., Schaub, M. T., Jadbabaie, A., and Kleinberg, J. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 115(48):E11221–E11230, 2018.

Bühler, T. and Hein, M. Spectral clustering based on the graph p-Laplacian. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 81–88, 2009.

Chan, T.-H. H. and Liang, Z. Generalizing the hypergraph laplacian via a diffusion process with mediators. *Theoretical Computer Science*, 806:416–428, 2020.

Chan, T.-H. H., Louis, A., Tang, Z. G., and Zhang, C. Spectral properties of hypergraph laplacian and approximation algorithms. *Journal of the ACM*, 65(3):1–48, 2018.

Chien, E., Pan, C., Peng, J., and Milenkovic, O. You are AllSet: A Multiset Function Framework for Hypergraph Neural Networks. *International Conference on Learning Representations (ICLR)*, 2022.

Chin, A., Chen, Y., M. Altenburger, K., and Ugander, J. Decoupled smoothing on graphs. In *The World Wide Web Conference*, pp. 263–272, 2019.

Dong, Y., Sawin, W., and Bengio, Y. Hnhn: Hypergraph networks with hyperedge neurons. *ICML Graph Representation Learning and Beyond Workshop*, 2020. URL https://arxiv.org/abs/2006.12278.

Elmoataz, A., Lezoray, O., and Bougleux, S. Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing. *IEEE transactions on Image Processing*, 17(7):1047–1060, 2008.

Feng, Y., You, H., Zhang, Z., Ji, R., and Gao, Y. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3558–3565, 2019.

Gleich, D. F. and Mahoney, M. W. Using local spectral methods to robustify graph-based learning algorithms. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 359–368, 2015.

Hein, M., Setzer, S., Jost, L., and Rangapuram, S. S. The total variation on hypergraphs - learning on hypergraphs revisited. In *Advances in Neural Information Processing Systems*, pp. 2427–2435, 2013.

Huang, J. and Yang, J. UniGNN: A unified framework for graph and hypergraph neural networks. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

Huang, Q., He, H., Singh, A., Lim, S.-N., and Benson, A. R. Combining label propagation and simple models out-performs graph neural networks. *International Conference on Learning Representations (ICLR)*, 2020.

Ibrahim, R. and Gleich, D. Nonlinear diffusion for community detection and semi-supervised learning. In *The World Wide Web Conference*, pp. 739–750, 2019.

Jia, J. and Benson, A. R. A unifying generative model for graph learning algorithms: Label propagation, graph convolutions, and combinations. *SIAM Journal on Mathematics of Data Science*, 4(1):100–125, 2022.

Juan, D.-C., Lu, C.-T., Li, Z., Peng, F., Timofeev, A., Chen, Y.-T., Gao, Y., Duerig, T., Tomkins, A., and Ravi, S. Ultra fine-grained image semantic embedding. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 277–285, 2020.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Klicpera, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized PageRank. In *International Conference on Learning Representations (ICLR)*, 2018.

Kyng, R., Rao, A., Sachdeva, S., and Spielman, D. A. Algorithms for lipschitz learning on graphs. In *Conference on Learning Theory*, pp. 1190–1223, 2015.

Li, P. and Milenkovic, O. Inhomogeneous hypergraph clustering with applications. *Advances in Neural Information Processing Systems*, 2017:2309–2319, 2017.

Li, P. and Milenkovic, O. Submodular hypergraphs: p-laplacians, cheeger inequalities and spectral clustering. In *International Conference on Machine Learning*, pp. 3014–3023. PMLR, 2018.

Li, P., He, N., and Milenkovic, O. Quadratic decomposable submodular function minimization: Theory and practice. *Journal of Machine Learning Research*, 21(106):1–49, 2020.

Liu, M., Veldt, N., Song, H., Li, P., and Gleich, D. F. Strongly local hypergraph diffusions for clustering and semi-supervised learning. In *Proceedings of the Web Conference 2021*, pp. 2092–2103, 2021.

Louis, A. Hypergraph markov operators, eigenvalues and approximation algorithms. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pp. 713–722, 2015.

Mallat, S. *A wavelet tour of signal processing*. Elsevier, 1999.

McPherson, M., Smith-Lovin, L., and Cook, J. M. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

Millán, A. P., Torres, J. J., and Bianconi, G. Explosive higher-order Kuramoto dynamics on simplicial complexes. *Physical Review Letters*, 124(21):218301, 2020.

Müller, R., Kornblith, S., and Hinton, G. E. When does label smoothing help? In *Advances in Neural Information Processing Systems*, pp. 4696–4705, 2019.

Neuhäuser, L., Lambiotte, R., and Schaub, M. T. Consensus dynamics and opinion formation on hypergraphs. In *Higher-Order Systems*, pp. 347–376. Springer, 2022.

Newman, M. E. Assortative mixing in networks. *Physical review letters*, 89(20):208701, 2002.

Rao, S., van der Schaft, A., and Jayawardhana, B. A graph-theoretical approach for the analysis and model reduction of complex-balanced chemical reaction networks. *Journal of Mathematical Chemistry*, 51(9):2401–2422, 2013.

Rossi, R. and Ahmed, N. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Saito, S., Mandic, D. P., and Suzuki, H. Hypergraph p-Laplacian: A differential geometry view. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Schaub, M. T., O'Clery, N., Billeh, Y. N., Delvenne, J.-C., Lambiotte, R., and Barahona, M. Graph partitions and cluster synchronization in networks of oscillators. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 26(9): 094821, 2016.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Srinivasan, B., Zheng, D., and Karypis, G. Learning over families of sets-hypergraph representation learning for higher order tasks. In *SIAM International Conference on Data Mining (SDM)*, pp. 756–764. SIAM, 2021.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Torres, L., Blevins, A. S., Bassett, D., and Eliassi-Rad, T. The why, how, and when of representations for complex systems. *SIAM Review*, 63(3):435–485, 2021.

Tudisco, F. and Hein, M. A nodal domain theorem and a higher-order Cheeger inequality for the graph p-Laplacian. *Journal of Spectral Theory*, 8(3):883–909, 2018.

Tudisco, F. and Higham, D. J. Node and edge nonlinear eigenvector centrality for hypergraphs. *Communications Physics*, 4(1):1–10, 2021.

Tudisco, F., Benson, A. R., and Prokopchik, K. Nonlinear higher-order label spreading. In *Proceedings of the Web Conference*, 2021.

Van der Schaft, A., Rao, S., and Jayawardhana, B. A network dynamics approach to chemical reaction networks. *International Journal of Control*, 89(4):731–745, 2016.

Veldt, N., Benson, A. R., and Kleinberg, J. Minimizing localized ratio cut objectives in hypergraphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1708–1718, 2020.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *International Conference on Machine Learning (ICML)*, pp. 6861–6871. PMLR, 2019.

Yadati, N., Nimishakavi, M., Yadav, P., Nitin, V., Louis, A., and Talukdar, P. Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Advances in Neural Information Processing Systems*, 32: 1511–1522, 2019.

Zhang, C., Hu, S., Tang, Z. G., and Chan, T. H. Re-revisiting learning on hypergraphs: confidence interval and subgradient method. In *International Conference on Machine Learning*, pp. 4026–4034. PMLR, 2017.

Zhang, R., Zou, Y., and Ma, J. Hyper-sagnn: a self-attention based graph neural network for hypergraphs. In *International Conference on Learning Representations (ICLR)*, 2020a.

Zhang, S., Huang, Z., Zhou, H., and Zhou, Z. Sce: Scalable network embedding from sparsest cut. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Jul 2020b. doi: 10.1145/3394486.3403068. URL http://dx.doi.org/10.1145/3394486.3403068.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. Learning with local and global consistency. In *Advances in neural information processing systems*, pp. 321–328, 2004.

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

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., and Koutra, D. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7793–7804, 2020.

Zhu, X., Ghahramani, Z., and Lafferty, J. D. Semi-supervised learning using gaussian fields and harmonic functions. In *International Conference on Machine Learning (ICML)*, pp. 912–919, 2003.

A. On the convergence of the nonlinear diffusion.

Our classification method is based on the nonlinear diffusion process on the hypergraph described in (7) and (8). Unlike the linear case, where the long term behaviour of the discrete dynamical systems can be easily studied, when linear mappings are combined with nonlinearities, neither the existence nor the uniqueness of a limit point is obvious for the nonlinear discrete diffusion model (7). This makes the convergence result in Theorem 4.1 particularly remarkable. In fact, already in the vector case, it is easy to find non convergent normalized iterates of the type $\widetilde{x}_{k+1} = \mathcal{L}(x_k) + y$, $x_{k+1} = \widetilde{x}_{k+1}/\|\widetilde{x}_{k+1}\|$, or iterates with multiple fixed points. Consider, for example the two iterations

$$z_{k+1} = A(Az_k)^{1.5} + y\,, \qquad \text{and} \qquad \begin{cases} \widetilde{x}_{k+1} = A(Ax_k)^{1.5} + y \\ x_{k+1} = \widetilde{x}_{k+1}/\|\widetilde{x}_{k+1}\|_\infty \end{cases} \tag{13}$$

where the power is taken component-wise and where A and y are 3-dimensional and chosen as follows:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \qquad y = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \end{bmatrix}.$$

None of the two sequences in (13) converge for most starting points x_0. Figure 4 illustrates this issue by showing the behaviour of the three coordinates of the iteration x_k as in (13), for a random sampled starting point x_0, sampled from a uniform distribution in $[0, 1]^3$. We observed the same behaviour for the sequence z_k and for any starting point sampled this way.

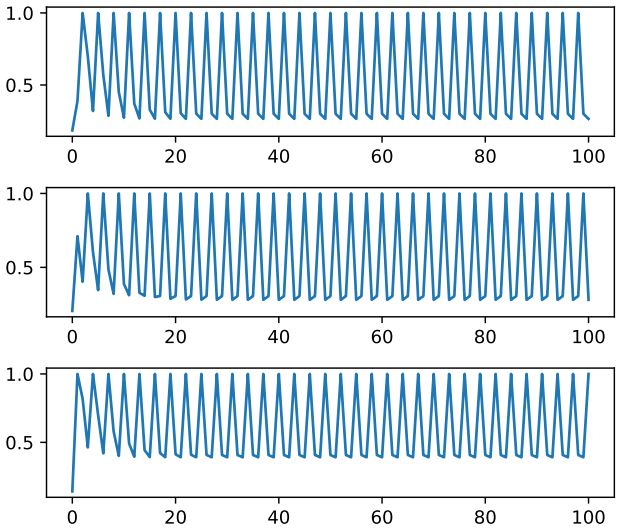

Figure 4: Example of a nonconvergent nonlinear diffusion. Each panel shows the behaviour of one of the three coordinates of x_k in (13) as a function of k, for $k = 1, \ldots, 100$.

B. Proof of Theorem 4.1.

Consider the iteration in (11). As $\sigma \in \hom_+(a)$ and $\varrho \in \hom_+(b)$ we have

$$\begin{aligned} \mathcal{L}(\lambda F) &= D^{-1/2}KW\sigma(K^\top \varrho(\lambda D^{-1/2}F)) = D^{-1/2}KW\sigma(\lambda^b K^\top \varrho(D^{-1/2}F)) \\ &= \lambda^{ab} D^{-1/2}KW\sigma(K^\top \varrho(D^{-1/2}F)) = \lambda \mathcal{L}(F) \end{aligned}$$

i.e. $\mathcal{L}$ is one-homogeneous. Moreover, as K and D are nonnegative matrices, $\mathcal{L} \in \hom_+(1)$. Similarly, as every node appears in at least one hyperedge, we see that under the assumptions on σ and ϱ it holds $\varphi(F) > 0$ for all $F > 0$. Thus, a similar computation as the one above shows that $\varphi(F) \in \hom_+(1)$. As a consequence, for any F with $\varphi(F) = 1$, every

component of $\mathcal{L}(F)$ is bounded and positive, i.e. there exist constants $M_{i,j} > 0$ such that

$$\max_{F:\varphi(F)=1} \mathcal{L}(F)_{ij} = \max_{F} \frac{\mathcal{L}(F)_{ij}}{\varphi(F)} \leq M_{ij} \,.$$

Hence, defining $M = \max_{ij} M_{ij} > 0$ we have that $\mathcal{L}(F) \leq M$ entrywise, for all F such that $\varphi(F) = 1$. As a consequence, since U is entrywise positive, there exists a constant $\widetilde{M} > 0$ such that $\mathcal{L}(F) \leq \widetilde{M}U$ entrywise, for all F such that $\varphi(F) = 1$. Using Theorem 3.1 in (Tudisco et al., 2021) we deduce that $F^{(k)} \to F_\star$ as $k \to \infty$ with $F_\star$ unique fixed point $F_\star = \alpha\mathcal{L}(F_\star) + (1-\alpha)U$ such that $\varphi(F_\star) = 1$ and $F_\star > 0$.

Now we show that $F_\star$ is also the only point where the gradient of

$$\widetilde{\ell}(F) := \left\| F - \frac{U}{\varphi(U)} \right\|^2 + \lambda \Omega_{\mu_{\sigma,\varrho}}(F)$$

vanishes. To this end, denoted by $S(F)$ is the $m \times (c+d)$ matrix of the hyperedge embedding $S(F) = \sigma(K^\top \varrho(F))$ and let $B(F) = KWS(F) = KW\sigma(K^\top \varrho(F))$. Notice that, with this notation, as observed in (4), we can write

$$\mu_{\sigma,\varrho}(\{f_i : i \in e\}) = S(F)_{e,:} \,.$$

As a consequence we get

$$\Omega_{\mu_{\sigma,\varrho}}(F) = \sum_{i \in V} \sum_{e: i \in e} w(e) \left\| (D^{-1/2}F)_{i,:} - \frac{1}{2} S(D^{-1/2}F)_{e,:} \right\|^2,$$

where, as it will be more convenient in the computation below, we are multiplying the μ_e term in the loss by $1/2$. Of course we can always do this by rescaling one of the two mappings σ or ϱ by a factor two, without losing any of their relevant properties nor generality in the proof. Thus, we have

$$\begin{aligned}
\Omega_{\mu_{\sigma,\varrho}}(D^{1/2}F) &= \sum_i \sum_{e: i \in e} w(e) \sum_j (F_{ij} - \frac{1}{2} S(F)_{ej})^2 \\
&= \sum_i \sum_{e: i \in e} w(e) \sum_j (F_{ij}^2 - F_{ij} S(F)_{ej}) + \frac{1}{4} \sum_i \sum_{e: i \in e} \sum_j w(e) S(F)_{ej}^2 \\
&= \sum_i \sum_j F_{ij}^2 \delta_i - F_{ij} B(F)_{ij} + \varphi(D^{1/2}F)^2 \\
&= \langle F, DF - B(F) \rangle + \varphi(D^{1/2}F)^2 \,,
\end{aligned}$$

where $\langle \cdot, \cdot \rangle$ denotes the matrix Frobenius scalar product. Therefore, it holds

$$\Omega_{\mu_{\sigma,\varrho}}(F) - \varphi(F)^2 = \langle F, F - D^{-1/2} B(D^{-1/2}F) \rangle = \langle F, F - \mathcal{L}(F) \rangle \,.$$

As $\mathcal{L}$ is 1-homogeneous and differentiable, by the Euler theorem for homogeneous functions we have that

$$\frac{d}{dF} \{ \Omega_{\mu_{\sigma,\varrho}}(F) - \varphi(F)^2 \} = \frac{d}{dF} \langle F, F - \mathcal{L}(F) \rangle = 2(F - \mathcal{L}(F)) \,.$$

Thus,

$$\frac{d}{dF} \{ \widetilde{\ell}(F) - \lambda \varphi(F)^2 \} = 2(F - U/\varphi(U) + \lambda(F - \mathcal{L}(F)) = 2((1+\lambda)F - \lambda\mathcal{L}(F) - U/\varphi(U))$$

which shows that the gradient of $\widetilde{\ell}(F) - \lambda\varphi(F)^2$ vanishes on a point $F_\star$ if and only if $F_\star$ is such that

$$F_\star = \frac{\lambda}{1+\lambda} \mathcal{L}(F_\star) + \frac{1}{1+\lambda} \frac{U}{\varphi(U)}$$

i.e. $F_\star$ is a fixed point of (11) for $\lambda = \alpha/(1-\alpha)$ and $U = U/\varphi(U)$. Finally, as the two loss functions $\widetilde{\ell}(F)$ and $\widetilde{\ell}(F) - \lambda\varphi(F)$ have the same minimizers on the slice $\{F : \varphi(F) = 1\}$, we conclude.

C. Additional results.

We present here additional performance results on the real-world datasets and the baseline methods considered in §5. In particular, we report mean precision and mean recall with their deviation of all the methods considered.

Table 3: Precision (mean and standard deviation) over five random samples of the training nodes $\mathcal{T}$. We compare HyperND and the five baseline methods (APPNP, HGNN, HyperGCN, SGC, SCE). Overall, HyperND performs better than the others.

	Method	**HOLS**	**APPNP**	**HGNN**	**HyperGCN**	**SGC**	**SCE**	**HTV**
Data	% labeled							
Citeseer	4.2%	65.02 ±0.98	56.76 ±1.92	56.05 ±2.29	**67.94 ±9.56**	54.75 ±8.74	55.66 ±1.45	41.58 ±1.5
Cora-author	5.2%	**74.81 ±1.66**	68.15 ±3.12	58.62 ±1.89	67.72 ±1.39	60.7 ±21.98	70.72 ±1.02	66.72 ±2.42
Cora-cit	5.2%	**81.44 ±2.18**	80.74 ±2.2	58.17 ±1.64	74.19 ±3.44	37.88 ±7.31	79.25 ±1.3	43.38 ±1.2
DBLP	4.0%	87.88 ±0.28	**88.3 ±0.14**	71.45 ±1.75	85.68 ±0.18	61.34 ±0.29	86.84 ±0.3	60.57 ±2.05
Foodweb	5.0%	**65.69 ±19.92**	58.73 ±6.85	60.04 ±21.68	54.61 ±5.58	57.84 ±4.49	47.54 ±8.6	59.93 ±3.57
Pubmed	0.8%	81.52 ±2.73	**81.63 ±0.74**	71.78 ±1.68	76.73 ±1.06	67.62 ±1.4	77.67 ±2.04	64.58 ±7.59

Table 4: Recall (mean and standard deviation) over five random samples of the training nodes $\mathcal{T}$. We compare HyperND and the five baseline methods (APPNP, HGNN, HyperGCN, SGC, SCE). Overall, HyperND performs better than the others.

	Method	**HyperND**	**APPNP**	**HGNN**	**HyperGCN**	**SGC**	**SCE**	**HTV**
Data	% labeled							
Citeseer	4.2%	**66.52 ±2.29**	58.64 ±1.22	60.38 ±2.04	48.22 ±11.8	66.46 ±5.95	55.96 ±1.83	51.23 ±1.73
Cora-author	5.2%	**76.76 ±2.81**	70.13 ±1.65	62.92 ±1.46	63.57 ±1.39	55.08 ±14.43	69.92 ±2.62	59.58 ±1.15
Cora-cit	5.2%	**82.56 ±0.74**	81.47 ±1.85	61.43 ±3.12	63.91 ±2.64	71.23 ±6.86	78.98 ±2.05	64.07 ±2.21
DBLP	4.0%	**88.52 ±0.11**	88.41 ±0.23	74.0 ±0.75	70.99 ±0.22	77.42 ±0.03	87.32 ±0.3	73.85 ±1.27
Foodweb	5.0%	62.94 ±10.37	**73.85 ±9.54**	51.05 ±17.92	44.44 ±7.4	57.45 ±0.47	57.77 ±17.33	49.48 ±1.03
Pubmed	0.8%	**81.21 ±1.8**	80.49 ±1.43	72.9 ±0.75	78.88 ±1.43	57.51 ±5.94	77.52 ±1.74	58.93 ±1.91