Nonlinear Higher-Order Label Spreading

Francesco Tudisco
Gran Sasso Science Institute
School of Mathematics
67100 L'Aquila, Italy
francesco.tudisco@gssi.it

Austin R. Benson
Cornell University
Department of Computer Science
Ithaca, NY 14853, USA
arb@cs.cornell.edu

Konstantin Prokopchik
Gran Sasso Science Institute
School of Computer Science
67100 L'Aquila, Italy
konstantin.prokopchik@gssi.it

ABSTRACT

Label spreading is a general technique for semi-supervised learning with point cloud or network data, which can be interpreted as a diffusion of labels on a graph. While there are many variants of label spreading, nearly all of them are linear models, where the incoming information to a node is a weighted sum of information from neighboring nodes. Here, we add nonlinearity to label spreading via nonlinear functions involving higher-order network structure, namely triangles in the graph. For a broad class of nonlinear functions, we prove convergence of our nonlinear higher-order label spreading algorithm to the global solution of an interpretable semi-supervised loss function. We demonstrate the efficiency and efficacy of our approach on a variety of point cloud and network datasets, where the nonlinear higher-order model outperforms classical label spreading, hypergraph clustering, and graph neural networks.

KEYWORDS

semi-supervised learning, hypergraphs, Laplacians, label spreading, label propagation, higher-order networks

ACM Reference Format:
Francesco Tudisco, Austin R. Benson, and Konstantin Prokopchik. 2021. Nonlinear Higher-Order Label Spreading. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3450035

1 INTRODUCTION

Label Spreading (LS) is a general algorithmic technique for Semi-Supervised Learning (SSL), where one infers unknown labels from known labels by iteratively *diffusing* or *spreading* the known labels over a similarity graph where nodes correspond to data points and edges connect similar data points [66]. Typically, the number of labeled data points is small (often less than 1% of the points), corresponding to settings where it is generally difficult to obtain labels. With generic point cloud data, edges are typically based on k-nearest-neighbors or ϵ-neighbors [30, 62, 65], but LS can also be used directly on relational data coming from, e.g., social networks [15], web graphs [34], or co-purchases [23]. A prototypical algorithm for LS is the local and global consistency approach of Zhou et al. [62]. In this method, all nodes iteratively spread their label values to neighbors, encouraging smoothness in label assignment over the data, while the values on labeled nodes are kept close to their initial label assignment.

The linearity of the diffusion makes LS simple to implement and analyze. From the perspective of an unlabeled data point, the corresponding node in the graph iteratively updates its label based on a fixed *linear* combination of the current labels of its neighbors. One can analyze the limiting behavior of this iterative process, which coincides with the solution of a Laplacian linear system and the minimization of a regularized loss function. At the same time, nonlinear methods provide stronger modeling capabilities, and are key to deep learning generally [24] and modern graph-based semi-supervised learning methods such as graph neural networks [26, 64]. However, we have limited theoretical understanding of graph neural networks, and they are generally difficult to interpret and challenging to scale to large datasets. Furthermore, such methods require substantial validation sets for hyperparameter tuning and are not particularly suited to settings in which there are a small number of labels.

Here, we propose a nonlinear label spreading method that lies in between these two approaches, which has more expressive modeling capability while remaining interpretable and scalable with theoretical guarantees on convergence. Our approach is based on two main ideas. First, we incorporate "higher-order" relationships between the data points, i.e., information about groups of nodes instead of just the similarities encoded by edges in a graph; this idea has recently been shown to improve label spreading, label propagation, and diffusion methods [18, 29, 39, 60]. Second, we devise a spreading process where a given node u updates its label based on the labels of the other nodes in its higher-order neighborhood, where the label information is "aggregated" with a nonlinear *mixing function* at each hyperedge containing u. We call this method *Nonlinear Higher-Order Label Spreading* (NHOLS).

With the additional nonlinear term, we obtain a method that maintains the same simplicity and scalability as the classical linear method, but with a much broader modeling power. Although the nonlinear term makes the analysis of the spreading process more challenging, we show that our method enjoys two fundamental properties similar to classical LS [62], for a broad class of nonlinear functions. First, the NHOLS process corresponds to running gradient descent on an interpretable objective function that decomposes into a function that imposes smoothness over the graph and a function that encourages the predictions to remain close to the initial assignment on labeled nodes. Second, NHOLS globally converges to a unique global minimizer of this objective.

Our analysis is based on interpreting the iterations of NHOLS as a type of tensor contraction. From this viewpoint, our convergence results are based on extensions of recent nonlinear Perron-Frobenius

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3450035

theory [21, 22]. These results also apply to a more general class of nonlinear iterations, and we use this to provide theoretical guarantees for a variant of recently proposed nonlinear diffusions that does not use higher-order information [28]. Furthermore, the special case of a linear mixing function essentially reduces to a recently proposed higher-order label spreading method [18].

In terms of implementation, NHOLS shares the same simplicity and efficiency as standard LS. Each iteration of the algorithm only requires a single pass over the input data, making it highly scalable. (In practice, though, this may involve a pre-processing to find all of the higher-order data, such as all of the triangles in a graph; this is typically fast for real-world networks [10, 35]). In numerical experiments, we show that the running time of NHOLS is orders of magnitude faster than graph neural network approaches, while also being nearly as fast as standard LS.

We also evaluate the predictive performance of NHOLS on a number of synthetic and real-world datasets, comparing against standard label spreading, hypergraph semi-supervised learning methods, and graph neural networks. We find that incorporating nonlinearities of higher-order information into label spreading almost always achieves the highest accuracies and that, in our setting, NHOLS is far superior to standard graph neural networks.

1.1 Additional related work

A key idea in many recent graph-based learning methods is that incorporating *higher-order* interactions involving multiple nodes can make large changes in performance. This has yielded improvements in numerous settings, including unsupervised clustering [9, 40, 52], localized clustering [38, 60], representation learning [47], link prediction [5, 8, 48], graph classification [2], ranking [5–7, 16], and data visualization [46]. A higher-order version of label spreading has also recently been developed for relational data [18], and this correspond to the special case of linear mixing functions within our framework.

There are also many machine learning methods for hypergraph data, and a standard approach is to first reduce the hypergraph to a graph upon which a graph-based method can be employed [1, 19, 40, 49, 61, 63]. These techniques are called *clique expansions*, as they place a (possibly weighted) clique in the graph for every hyperedge. Using a linear mixing function in our framework is a clique expansion technique, which we cover in Section 3. However, our analysis focuses on nonlinear mixing functions, which do not correspond to clique expansions. Thus, our framework is conceptually closer to hypergraph methods that avoid clique expansions, such as those based on nonlinear Hypergraph Laplacian operators [12, 41, 57] or generalized splitting functions [53, 54].

There are nonlinear graph-based semi-supervised learning techniques that use p-Laplacians [3, 11, 28, 34] or p-norms [58]. More broadly, our framework follows several recent graph methods that add interpretable nonlinearities onto traditional linear methods to improve expressibility and practical performance [13, 14, 28]

2 BACKGROUND ON STANDARD LABEL SPREADING

We first review a standard label spreading (LS) technique that is essentially the same as the one of Zhou et al. [62] so that we can later draw parallels with our proposed nonlinear higher-order method. Let $G = (V, E, \omega)$ be a weighted undirected graph with nodes $V = \{1, \dots, n\}$, edge set $E \subseteq V \times V$ and edge-weight function $\omega(ij) > 0$. As mentioned in the introduction, G typically represents either a similarity graph for a point cloud or a relational network. Let A be the adjacency matrix of G, i.e., $A_{ij} = \omega(ij)$ if $ij \in E$ and $A_{ij} = 0$ otherwise. Furthermore, let $D_G = \mathrm{Diag}(d_1, \dots, d_n)$ be the diagonal degree matrix of G, where $d_i = \sum_j A_{ij}$. Throughout this paper, we will assume that G has no isolated nodes, that is D_G has no zero diagonal entries. (This is a standard assumption and, in practice, if node i is isolated, we can either remove it from the graph or add a self-loop ii; we do not run into isolated nodes in our experiments.) Finally, let

$$S = D_G^{-1/2} A D_G^{-1/2} \tag{1}$$

be the normalized adjacency matrix.

Our goal is to provide a label in $\{1, \dots, c\}$ to each node, and we know the label of (usually a small) subset of the nodes. The initial labels are represented by membership vectors in an $n \times c$ matrix Y, where $Y_{i,\ell} = 1$ if node i has initial label ℓ and $Y_{i,\ell} = 0$ otherwise. Given an initial guess $F^{(0)} \in \mathbb{R}^{n \times c}$, the label spreading algorithm iteratively computes

$$F^{(r+1)} = \beta S F^{(r)} + \gamma Y \qquad r = 0, 1, 2, \dots, \tag{2}$$

with $\beta, \gamma \geq 0$ and $\beta + \gamma = 1$. The iterates converge to the solution of the linear system $(I - \beta S)F^* = \gamma Y$, but in practice a few iterations of (2) with the initial point $F^{(0)} = Y$ suffices [20]. This yields an approximate solution $\widetilde{F}^*$. The prediction on an unlabeled node j is then $\arg\max_\ell \widetilde{F}^*_{j,\ell}$. This setup is equivalent to performing label spreading once per class with an initial binary vector y (i.e., one vector y for each column of Y).

When initialized with $F^{(0)} = 0$ (which produces $F^{(1)} = \gamma Y$), the label spreading procedure in (2) is naturally interpreted as diffusing the input labels Y along the edges of the graph by means of the adjacency matrix S. At the same time, this method can also be interpreted as gradient descent applied to a quadratic regularized loss function and as a discrete dynamical system that spreads the initial value condition $F^{(0)} = Y$ through the graph via a linear gradient flow. We briefly review these formulations. Let ψ be the quadratic energy loss function that is separable on the columns of F,

$$\psi(F) = \sum_{\ell=1}^{c} \psi_\ell(F_{:,\ell}) = \sum_{\ell=1}^{c} \frac{1}{2} \left\{ \|F_{:,\ell} - Y_{:,\ell}\|_2^2 + \lambda\, F_{:,\ell}^\top \Delta F_{:,\ell} \right\}, \tag{3}$$

where $\Delta = I - D_G^{-1/2} A D_G^{-1/2} = I - S$ is the normalized Laplacian, and $\lambda > 0$ is a regularization parameter. This loss encourages smoothness over the graph (via the normalized Laplacian term) while also keeping the solution on labeled nodes close to the initial label assignment.

For $\ell \in \{1, \dots, c\}$, consider the the dynamical system

$$\frac{\mathrm{d}}{\mathrm{d}t} f(t) = -\nabla \psi_\ell(f(t)), \qquad f(0) = Y_{:,\ell}\,. \tag{4}$$

Since ψ_ℓ is convex, $\lim_{t\to\infty} f(t) = f^*$ such that $\psi_\ell(f^*) = \min_f \psi_\ell(f)$. Label spreading in (2) coincides with gradient descent applied to (3) or, equivalently, with explicit Euler integration applied to (4),

for a particular value of the step length h. In particular,

$$f - h\nabla\psi_\ell(f) = f - h(f - Y_{:,\ell} + \lambda\Delta f) = (1-h-h\lambda)f + h\lambda Sf + hY_{:,\ell}, \tag{5}$$

which, for $(1-h)/h = \lambda$, coincides with one iteration of (2) applied to the ℓth column of F. Moreover, as $F^{(r)} \geq 0$ for all r, this gradient flow interpretation shows that the global minimizer of (3) is nonnegative, i.e., $\min_F \psi(F) = \min_{F\geq 0} \psi(F)$. In the next section, we use similar techniques to derive our nonlinear higher-order label spreading method.

3 NONLINEAR HIGHER-ORDER LABEL SPREADING

Now we develop our nonlinear higher-order label spreading technique. We assume that we have a 3-uniform hypergraph $H = (V, \mathcal{E}, \tau)$ capturing higher-order information on the same set of nodes as the weighted graph $G = (V, E, \omega)$, where $\mathcal{E} \subseteq V \times V \times V$ and τ is a hyperedge weight function with $\tau(ijk) > 0$ for $ijk \in \mathcal{E}$. In our experiments, we will usually derive H from G by considering the hyperedges of H to be the set of triangles (i.e., 3-cliques) of G. However, in principle we could use any hypergraph. We also do not need the associated graph G, but we keep it for greater generality and find it useful in practice.

Below, we develop our methodology for 3-regular hypergraphs for simplicity, but our ideas generalize to arbitrary hypergraphs. More specifically, our convergence results in Theorem 3.1 only assume that we iteratively apply a nonlinear function with certain properties, which has no dependence on the higher-order structure. And in terms of the practical implementation, we use nonlinear mixing functions that take the p-mean of values on nodes in a hyperedge before spreading back to each node in the hyperedge, and this can be done with heterogeneous hypergraphs that contain hyperedges of various sizes. Finally, the interpretable objective function that our method optimizes in (13) has smoothness terms for 2-way and 3-way relationships among nodes; including hyperedges of larger sizes just corresponds to additional smoothness terms.

3.1 Nonlinear Second-order Label Spreading with Mixing Functions

We represent H via the associated adjacency tensor $\mathcal{A}$, defined by $\mathcal{A}_{ijk} = \tau(ijk)$ if $ijk \in \mathcal{E}$ and $\mathcal{A}_{ijk} = 0$ otherwise. Analogous to the graph case, let $D_H = \mathrm{Diag}(\delta_1, \ldots, \delta_n)$ be the diagonal matrix of the hypergraph node degrees, where $\delta_i = \sum_{j,k:\, ijk\in\mathcal{E}} \tau(ijk) = \sum_{jk} \mathcal{A}_{ijk}$. Again, we assume that H has no isolated nodes so that $\delta_i > 0$ for all $i \in V$. As in the graph case, this is a relatively standard assumption. In practice, if node $\delta_i = 0$, then we compensate by using the information available on the edges (i.e., in G): for all neighbors j of i in G we add the hyperedge iji to H. This is a higher-order analog of the addition of a self-loop for isolated nodes in graphs.

As noted in the introduction, we will make use of nonlinear *mixing functions*, which we denote by $\sigma\colon \mathbb{R}^2 \to \mathbb{R}$. For a tensor $T = T_{ijk}$, we define the tensor map $T\sigma\colon \mathbb{R}^n \to \mathbb{R}^n$ entrywise:

$$T\sigma(f)_i = \textstyle\sum_{jk} T_{ijk}\, \sigma(f_j, f_k). \tag{6}$$

Algorithm 1: NHOLS: Nonlinear Higher-Order Label Spreading.

Input: Tensor $\mathcal{A}$; matrix A; mixing function $\sigma : \mathbb{R}^2 \to \mathbb{R}$; label matrix $Y \in \{0,1\}^{n\times c}$; scalars $\alpha, \beta, \gamma \geq 0$ with $\alpha + \beta + \gamma = 1$; smoothing parameter $0 < \varepsilon < 1$; stopping tolerance tol

Output: Predicted labels $\hat{y} \in \{1, \ldots, c\}^n$

```
F̃ ∈ ℝ^{n×c}   // Store approximate solutions
for ℓ = 1, ..., L do
    // Initialize with label smoothing
    y_ε ← (1 − ε)Y_{:,ℓ} + ε𝟙
    f^(0) ← y_ε
    repeat
        // Follow (8) and (9)
        g ← αS(f^(r)) + βSf^(r) + γy_ε
        f^(r+1) ← g/φ(g)
    until ‖f^(r+1) − f^(r)‖/‖f^(r+1)‖ < tol
    F̃_{:,ℓ} ← f^(r+1)
end
for i = 1, ..., n do ŷ_i = arg max_ℓ F̃_{i,ℓ}
```

Hence, in analogy with the matrix case, we denote by $\mathcal{S}\colon \mathbb{R}^n \to \mathbb{R}^n$ the *nonlinear normalized adjacency tensor* map

$$\mathcal{S}(f) = D_H^{-1/2}\mathcal{A}\,\sigma(D_H^{-1/2}f). \tag{7}$$

This nonlinear tensor contraction will be the basis for our method.

We need one additional piece of notation that is special to the higher-order case, which is a type of energy function that will be used to normalize iterates in order to guarantee convergence. Let B be the matrix with entries $B_{ij} = \sum_k \mathcal{A}_{kij}$. Define $\varphi\colon \mathbb{R}^n \to \mathbb{R}$ by

$$\varphi(f) = \tfrac{1}{2}\sqrt{\textstyle\sum_{ij} B_{ij}\, \sigma\Big(\frac{f_i}{\sqrt{\delta_i}}, \frac{f_j}{\sqrt{\delta_j}}\Big)^2}. \tag{8}$$

Finally, we arrive at our nonlinear higher-order label spreading (NHOLS) method. Given an initial vector $f^{(0)} \in \mathbb{R}^n$, we define the NHOLS iterates by

$$\begin{cases} g^{(r)} = \alpha\mathcal{S}(f^{(r)}) + \beta Sf^{(r)} + \gamma y, \\ f^{(r+1)} = g^{(r)}/\varphi(g^{(r)}) \end{cases} \qquad r = 0, 1, 2, \ldots \tag{9}$$

where $\alpha, \beta, \gamma \geq 0$, $\alpha+\beta+\gamma = 1$, and y is an initial label membership vector. Provided that σ is positive (i.e., $\sigma(a, b) > 0$ for any $a, b > 0$) and the initial vector $f^{(0)}$ is nonnegative, then all iterates are nonnegative. This assumption on the mixing function will be crucial to prove the convergence of the iterates. We perform this iteration with one initial vector per label class, analogous to standard Label Spreading. Algorithm 1 gives the overall procedure.

We now discuss several important properties of Algorithm 1. The parameters α, β, γ are a convex combination of three terms and allow us to tune the contributions of the first-order (graph) and second-order (hypergraph) smoothness terms; we will show this derivation in the next section. For $\beta = 0$, we obtain a purely second-order method, which can be useful when we do not have access to first-order data (e.g., we only have a hypergraph). The case of $\alpha = 0$ reduces to a normalized version of the standard LS as

in (2). Algorithm 1 also uses label smoothing in the initialization (the parameter ε), which will be useful for proving convergence results and can also improve generalization [45].

We can compute the iteration in (9) efficiently — each iteration requires one matrix-vector product and one "tensor-martix" product, which only takes a single pass over the input data. Therefore, NHOLS scales linearly with the number of edges and hyperedges, i.e., its computational cost is linear in the size of the data. A common setup for our experiments is that $\mathcal{A}$ is an adjacency tensor corresponding to all of the triangles in the graph. In this case, the cost per iteration is linear in the number of triangles. More formally, the running time of NHOLS in this case is $O(c(m+t))$, where c is the number of classes, m is the number of edges, and t is the number of triangles. On many real-world datasets, the number of triangles is still roughly linear in the number of edges [10], which would make each iteration have $O(cm)$ runtime.

The special case of a linear mixing function. The mixing function σ is responsible for the nonlinearity of the method. The linear mixing function $\sigma(a,b) = a + b$ reduces NHOLS to an approach based on a clique expansion graph, which, for $\beta = 0$, corresponds to a normalized version of the approaches from Eswaran et al. [18] and Zhou et al. [63]. To see this, let K be the $n \times |\mathcal{E}|$ incidence matrix of the hypergraph H, where $K_{i,e} = 1$ if node i is in the hyperedge e and $K_{i,e} = 0$ otherwise. Furthermore, let W be the diagonal matrix of hyperedge weights $\tau(e)$, $e \in \mathcal{E}$. Then $2\big(KWK^\top\big)_{ij} = \sum_k \mathcal{A}_{ijk} + \mathcal{A}_{ikj}$, and for $\sigma(a,b) = a + b$, we have

$$\begin{aligned}\mathcal{S}(f)_i &= \delta_i^{-1/2} \sum_{jk} \mathcal{A}_{ijk}\delta_j^{-1/2} f_j + \mathcal{A}_{ijk}\delta_k^{-1/2} f_k \\ &= \delta_i^{-1/2} \sum_j \Big(\sum_k \mathcal{A}_{ijk} + \mathcal{A}_{ikj}\Big)\delta_j^{-1/2} f_j = \big(\Theta f\big)_i,\end{aligned}$$

where $\Theta = 2D_H^{-1/2} KWK^T D_H^{-1/2}$ is the normalized adjacency matrix of the clique expansion graph [63].

3.2 Global convergence and optimization framework

Our NHOLS method extends standard LS in a natural way. However, with the nonlinear mixing function σ, it is unclear if the iterates even converge or to what they might converge. In this section, we show that, remarkably, NHOLS is globally convergent for a broad class of mixing functions and is minimizing a regularized objective similar to the one in (3) for standard LS. Proofs of the theoretical results in this section are in the appendix.

For convergence, we only require the mixing function to be positive, order-preserving, and homogeneous. Recall that these three properties for a general function $\Phi\colon \mathbb{R}^n \to \mathbb{R}^n$ are as follows.

$$\textit{Positivity}\colon\ \Phi(x) > 0 \text{ for all } x > 0. \tag{10}$$

$$\textit{Order-preserving}\colon\ \Phi(y) \geq \Phi(x) \text{ if } y \geq x. \tag{11}$$

$$p\textit{-homogeneity}\colon\ \Phi(cx) = c^p \Phi(x) \text{ for all } c > 0 \text{ and all } x. \tag{12}$$

(The inequalities above are interpreted entrywise.) We state below our main convergence theorem

Theorem 3.1. *Let $F\colon \mathbb{R}^n \to \mathbb{R}^n$ be positive, order-preserving, and p-homogeneous with $0 < p \leq 1$. Let $\varrho\colon \mathbb{R}^n \to \mathbb{R}$ be positive and 1-homogeneous, and let y be a positive vector. If there exists a $C > 0$ such that $F(f) \leq Cy$ for all f with $\varrho(f) = 1$, then, for any $f^{(0)} > 0$, the sequence*

$$g^{(r)} = F(f^{(r)}) + y \qquad f^{(r+1)} = g^{(r)}/\varrho(g^{(r)})$$

converges to a vector $f^* > 0$. Moreover, f^* is the unique fixed point of the mapping $F(f) + y$ such that $\varrho(f^*) = 1$.

Theorem 3.1 allows us to prove the convergence of NHOLS. To this end, we require an entry-wise positive label initialization. This is the reason for the smoothed membership vectors $y_\varepsilon = (1-\varepsilon)Y_{:,\ell} + \varepsilon\mathbb{1}$ in Algorithm 1. (In other words, $(y_\varepsilon)_i = (1-\varepsilon)Y_{i,\ell} + \varepsilon > 0$.) This assumption is not restrictive in practice as ε can be chosen fairly small, and we can also interpret this as a type of label smoothing [45, 50] (although similarly named, *label smoothing* is an entirely different concept than *label spreading*).

The following corollary shows that the NHOLS iterates converge for a broad class of mixing functions.

Corollary 3.2. *Let $f^{(r)}$ be the iterates in Algorithm 1. If σ is positive, order-preserving, and 1-homogeneous, then the sequence $\{f^{(r)}\}_r$ converges to a unique stationary point $f^* > 0$ with $\varphi(f^*) = 1$.*

In addition to global convergence, a fundamental property of the standard LS method is its interpretable optimization framework. We next show that for differentiable, 1-homogeneous mixing functions σ, the limit f^* of NHOLS minimizes a regularized objective that encourages smoothness over the labels assignment with respect to the graph and hypergraph interactions while also constraining the predictions on originally labeled nodes to be close the given labels. For a smoothed membership vector $y_\varepsilon = (1-\varepsilon)Y_{:,\ell} + \varepsilon\mathbb{1}$ with $0 < \varepsilon < 1$, consider the loss function

$$\begin{aligned}\vartheta(f) = \frac{1}{2}\Bigg\{ &\Big\| f - \frac{y_\varepsilon}{\varphi(y_\varepsilon)}\Big\|_2^2 + \lambda \sum_{ij} A_{ij}\Big(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\Big)^2 \\ &+ \mu \sum_{ijk} \mathcal{A}_{ijk}\Big(\frac{f_i}{\sqrt{\delta_i}} - \frac{1}{2}\sigma\Big(\frac{f_j}{\sqrt{\delta_j}}, \frac{f_k}{\sqrt{\delta_k}}\Big)\Big)^2\Bigg\}.\end{aligned} \tag{13}$$

As for the case of standard LS, ϑ has a fitting constraint and a smoothness component. However, there are two main differences. First, the fitting constraint component now considers a normalized membership vector $\widetilde{y}_\varepsilon = y_\varepsilon/\varphi(y_\varepsilon)$. As $\varphi(\widetilde{y}_\varepsilon) = 1$, we seek for a minimizer of ϑ in the slice $\{f : \varphi(f) = 1\}$. Second, the smoothness regularization now combines the graph Laplacian term with a new tensor-based term that encourages smoothness on the higher-order interactions among the nodes.

At a high level, the new tensor-based smoothness term encourages that, for all higher-order relationships ijk, we have that the value at node i is similar to a mixture of the values at j and k, the value at j is similar to a mixture of the values at i and k, and the value at k is similar to a mixture of the values at i and j. For our experiments, we will take σ to be various p-means so the mixture just corresponds to a generalized (nonlinear) mean, and we will see that certain choices of the mixing function, such as $\sigma(a,b) = 2\cdot\max(a,b)$, tend to work well in practice (the factor of 2 comes from the factor of $\frac{1}{2}$ in front of σ in (13)). The following theorem says that Algorithm 1 is optimizing this new objective.

Theorem 3.3. *Let $f^{(r)}$ be the sequence generated by Algorithm 1. If σ is positive, 1-homogeneous, and differentiable, then the sequence $\{f^{(r)}\}_r$ converges to the unique global solution of the constrained optimization problem*

$$\min \vartheta(f) \text{ s.t. } f > 0 \text{ and } \varphi(f) = 1 \tag{14}$$

with $\mu = \alpha/\gamma$ and $\lambda = \beta/\gamma$.

Analogous to standard label spreading, when started with $f^{(0)} = 0$, the iterations in (9) can be naturally interpreted as diffusing the input labels y_ε simultaneously along the edges and the hyperedges of G and H by means of the adjacency mappings S and $\mathcal{S}$, respectively. Moreover, just like standard LS, NHOLS can also be interpreted as projected gradient descent applied to a regularized loss function and as a projected diffusion process that spreads the input label assignment via a gradient flow. However, unlike the standard LS, the loss function in this setting is not convex and the gradient flow is no longer linear.

To this end, let $\widetilde{\vartheta}$ be the energy function

$$\widetilde{\vartheta}(f) = \vartheta(f) - \frac{\mu}{2}\varphi(f)^2$$

Note that ϑ and $\widetilde{\vartheta}$ have the same minimizing points on $\{f : \varphi(f) = 1\}$. Furthermore, consider the dynamical system

$$\frac{\mathrm{d}}{\mathrm{d}t} f(t) = -\nabla\widetilde{\vartheta}\big(f(t)\big), \quad f(0) = y_\varepsilon\,. \tag{15}$$

Then, our NHOLS iterates in (9) correspond to projected gradient descent applied to $\widetilde{\vartheta}$ or, equivalently, to a forward Euler method applied to (15). In fact, for y_ε such that $\varphi(y_\varepsilon) = 1$, we can show that

$$f - h\nabla\widetilde{\vartheta}(f) = (1 - h - \lambda h + \mu h)f + h\lambda Sf + h\mu\mathcal{S}(f) + hy_\varepsilon, \tag{16}$$

so choosing the step length h to satisfy $(1-h)/h = \lambda + \mu$ coincides with one step of (9). (A proof of this identity is in the appendix, within the proof of Theorem 3.3.) Unlike standard LS, the loss functions ϑ and $\widetilde{\vartheta}$ are not convex in general. Thus, the long-term behavior $\lim_{t\to\infty} f(t)$ of the dynamical system (15) is not straightforward. Despite this, Theorem 3.3 shows that just like its linear counterpart, NHOLS converges to a unique global minimizer of ϑ over the set of nonnegative vectors.

Finally, we note that, as $\nabla\psi$ in (4) can be interpreted as a discrete Laplacian operator on the graph [62], we can interpret $\nabla\widetilde{\vartheta}$, and thus $\mathcal{S}$, as a hypergraph Laplacian operator, which adds to the recent literature on (nonlinear) Laplacians on hypergraphs [12, 42].

3.3 Extensions

Theorem 3.1 makes it easy to extend the second-order setting discussed in this work to hyperedges of any order. For example, with size-4 hyperedges, we would use a nonlinear map

$$T\sigma(f)_i = \sum_{jkl} T_{ijkl}\sigma(f_j, f_k, f_l)$$

and define a spreading term using an adjacency tensor with four indices. With some additional algebra, we would get an objective function like the one in (13), just with an additional smoothness term for 4-way interactions.

The generality of Theorem 3.1 also makes it easy to design new types of nonlinear iterative methods. For example, nonlinearities could be added to the graph term, and there could be nonlinear mixing between interactions of different sizes. These are more expressive models, but our NHOLS method has the remarkable advantage of also being connected to the global optimum of an interpretable objective function. For the more general maps enabled by Theorem 3.1, the corresponding objective functions are unclear, although this is an opportunity for future research.

Table 1: Five mixing functions obtained from generalized means with different parameters s. By Corollary 3.2, Algorithm 1 converges for any choice of s.

mixing function	parameter s	$\sigma_s(a,b) = 2\left((a^s + b^s)/2\right)^{1/s}$
arithmetic	$s = 1$	$a + b$
harmonic	$s = -1$	$4\big(1/a + 1/b\big)^{-1}$
L^2	$s = 2$	$\sqrt{2(a^2 + b^2)}$
geometric	$s \to 0$	$2\sqrt{ab}$
maximum	$s \to \infty$	$2 \cdot \max(a, b)$

We consider one extension that is based on just incorporating nonlinearity into the graph method, which we will use in our experiments to show that higher-order information is helpful. A nonlinear first-order label spreading has the general form $f^{(r+1)} = \beta M\sigma(f^{(r)}) + (1-\beta)y_\varepsilon$, where $M\sigma(f)$ is the vector with entries $M\sigma(f)_i = \sum_j M_{ij}\sigma(f_j)$, σ is a nonlinear map, and M is a matrix associated with the graph. Ibrahim and Gleich [28] recently introduced one such nonlinear diffusion model, which is based on a forward Euler discretization of a first-order gradient flow similar to (4). The diffusion iterates (approximately) follow

$$f^{(r+1)} = \beta D_G^{-1} A(f^{(r)})^p + \gamma y_\varepsilon, \qquad r = 0, 1, 2, \ldots, \tag{17}$$

where the pth power is taken entry-wise, and p, β, γ are coefficients in the interval $[0, 1]$ with $\beta + \gamma = 1$. Since the iterations in (17) are unbounded, Ibrahim and Gleich [28] threshold each $f^{(r)}$ by setting to 0 and 1 all entries that are smaller than 0 or larger than 1, respectively. While practical, this procedure does not guarantee convergence. If we instead normalize each step with a vector norm, such as

$$g^{(r)} = \beta D_G^{-1} A(f^{(r)})^p + \gamma y_\varepsilon, \qquad f^{(r+1)} = g^{(r)}/\|g^{(r)}\|_1, \tag{18}$$

then Theorem 3.1 says that the sequence converges to a unique limit, for all choices of the initial value $f^{(0)} > 0$ and all $p \in [0, 1]$.

4 NUMERICAL EXPERIMENTS

We now perform experiments on both synthetic and real-world data. Our aim is to compare the performance of standard (first-order) label spreading algorithm with our second-order methods, using different mixing functions σ. Based on (13), a natural class of functions are the one-parameter family of generalized means scaled by factor 2, i.e., $\sigma_s(a,b) = 2\left((a^s + b^s)/2\right)^{1/s}$. We use the five generalized means in Table 1. The experiments can be reproduced using the code available at https://github.com/Doublelucker/NHOLS.

The function σ_s is 1-homogeneous, order-preserving, and positive for all values of $s \in \mathbb{R}$, including the limit cases of $s \in \{0, +\infty\}$. Thus, by Corollary 3.2, Algorithm 1 converges for any choice of s. The maximum function, however, is not differentiable and thus

we cannot prove that the computed limit of the sequence $f^{(r)}$ optimizes the loss function (13); however, one can approximate the maximum with a smooth function by choosing a large finite s, and Theorem 3.3 still applies. Also, as shown above, the case $s = 1$ (where σ_s is linear) essentially corresponds to a clique expansion method, so Algorithm 1 is nearly the same as existing methods based on clique expansions [18, 60, 63], where the adjacency matrix is a convex combination of the clique expansion graph induced by the tensor $\mathcal{A}$ and the graph A. In our experiments, we use at most 40 iterations within Algorithm 1, a stopping tolerance of 1e-5, and smoothing parameter $\varepsilon = 0.01$. Other settings of these parameters produced similar results.

We compare against several techniques, in addition to standard label spreading. One is hypergraph total variation (HTV) minimization, which is designed for clustering hypergraphs with larger hyperedges but is still applicable to our semi-supervised setup [27]. Another set of baselines comes from graph neural networks (GNNs). While such methods are lauded for their success on many graph learning tasks, our experiments are for a regime with a smaller number of labeled nodes than is typical for using GNNs. For GNNs, we use GraphSAGE [25] with two layers, ReLU activation, and max aggregation, as well as a Graph Convolutional Network (GCN) with two layers and ReLU activation. We also use Planetoid, a graph-based semi-supervised learning algorithm [59]. For the GNNs and Planetoid, we use data features if they are available; otherwise, we use the standard one-hot encodings of the nodes' identities as the features. Finally, we use the nonlinear first-order label spreading (NFOLS) in (17) with $p = 0.5$. None of the baselines incorporate both first- and second-order information in the graph, which is one advantage provided by NHOLS.

All of the algorithms that we use have hyperparameters. For the label spreading and HTV methods, we run 5-fold cross validation with label-balanced 50/50 splits over a small grid to choose these hyperparameters. For standard label spreading, the grid is $\beta \in \{0.1, 0.2, \ldots, 0.9\}$. For higher-order label spreadings, the grid is $\alpha \in \{0.3, 0.4, \ldots, 0.8\}$ and $\beta \in \{0.1, 0.25, 0.40, 0.55\}$ (subject to the constraint that $\alpha + \beta < 1$). HTV has a regularization term λ, for which we search over $\lambda = (1 - \beta)/\beta$ for $\beta \in \{0.1, 0.2, \ldots, 0.9\}$ (the same grid as LS). We choose the parameters that give the best mean accuracy over the five folds. The GNN approaches are much slower, so we split the labeled data into a training and validation sets with a label-balanced 50/50 split (a standard setup for training such models). We use the ADAM optimizer with default β parameters and search over learning rates $\eta \in \{0.01, 0.001, 0.0001\}$ and weight decays $\omega \in \{0, 0.0001\}$, using at most 15 epochs, with early stopping to limit overfitting (we stop once accuracy on the validation set exceeds 95%). Finally, for Planetoid, we manually tuned the number of embedding iterations to ensure convergence.

4.1 Synthetic benchmark data

We first compare the semi-supervised learning algorithms on synthetic graph data generated with the Stochastic Block Model (SBM). The SBM is a generative model for graph data with prescribed cluster structure. We use the variant of an SBM with two parameters — p_{in} and p_{out} — which designate the edge probabilities within the same label and across different labels, respectively. Generative block models are a common benchmark to test the performance of semi-supervised learning methods [31, 33, 43, 44]. Here we analyze the performance of different methods on random graphs drawn from the SBM where nodes belong to three different classes of size (number of nodes) 100, 200, and 400. We sample graphs for different values of the parameters p_{in} and p_{out}. We fix $p_{\text{in}} = 0.1$ and let $p_{\text{out}} = p_{\text{in}}/s$ for $s \in \{2.0, 2.5, 3.0, 3.5, 4.0\}$. With this setup, small values of s correspond to more difficult classification problems. We test the various algorithms for different percentages of known labels per class ($\{6\%, 9\%, 12\%, 15\%, 18\%, 21\%\}$).

The colored tables in Figure 1 show the mean clustering accuracy over ten random samples for each SBM setting and each percentage of input labels. We observe that the nonlinear label spreading methods perform better overall, with the maximum function performing the best in nearly all the cases. The performance gaps can be quite substantial. For example, when only 6% of the labels are given, NHOLS achieves up to 89% mean accuracy, while the baselines do not achieve greater than 81%.

Moreover, as discussed in Section 3.1, NHOLS scales linearly with the number of nonzero elements of $\mathcal{S}$ and S and thus is typically just slightly more expensive than standard LS. This is demonstrated by the results in Figure 2, where we compare mean execution time over ten runs for different methods. For NHOLS we show the mean execution time between the five choices of the mixing function listed in Table 1. We generate random SBM graphs with three labels of equal size, increasing the number of nodes n and edge probabilities $p_{\text{in}} = \log(n)^2/n$ with fixed membership probability ratios $p_{\text{out}} = p_{\text{in}}/3$. The times correspond to a single fixed set of values of the hyperparameters. We find that HTV is around one order of magnitude slower than NHOLS, while GCN and GraphSAGE are three to five orders of magnitude slower.

4.2 Real-world data

We also analyze the performance of our methods on six real-world datasets (Table 2). The first two datasets come from relational networks, namely Facebook friendship graphs of Rice University and Caltech from the Facebook100 collection [51]. The labels are the dorms in which the students live — there are 9 dorms for Rice and 8 for Caltech. Facebook friendships form edges, and the tensor entries correspond to triangles (3-cliques) in the graph. We preprocessed the graph data by first removing students with missing labels and then taking the largest connected component of the resulting graph. The GNNs use a one-hot encoding for features.

The next four graphs are derived from point clouds: optdigits [17, 56], pendigits [4, 17], MNIST [36], and Fashion-MNIST [55]. Each of these datasets has 10 classes, corresponding to one of 10 digits or to one of 10 fashion items. In these cases, we first create 7-nearest-neighbor graphs. Tensor entries correspond to triangles in this graph. We give the GNNs an additional advantage by providing node features derived from the data points. The optdigits and pendigits datasets come with several hand-crafted features, and we use the embedding given by the first 10 principal components for node features. The MNIST and Fashion-MNIST datasets are just raw images; here, we use an embedding from the first 20 principal components of the images as node features for the GNN. We also tried one-hot encodings and the raw data points as features instead

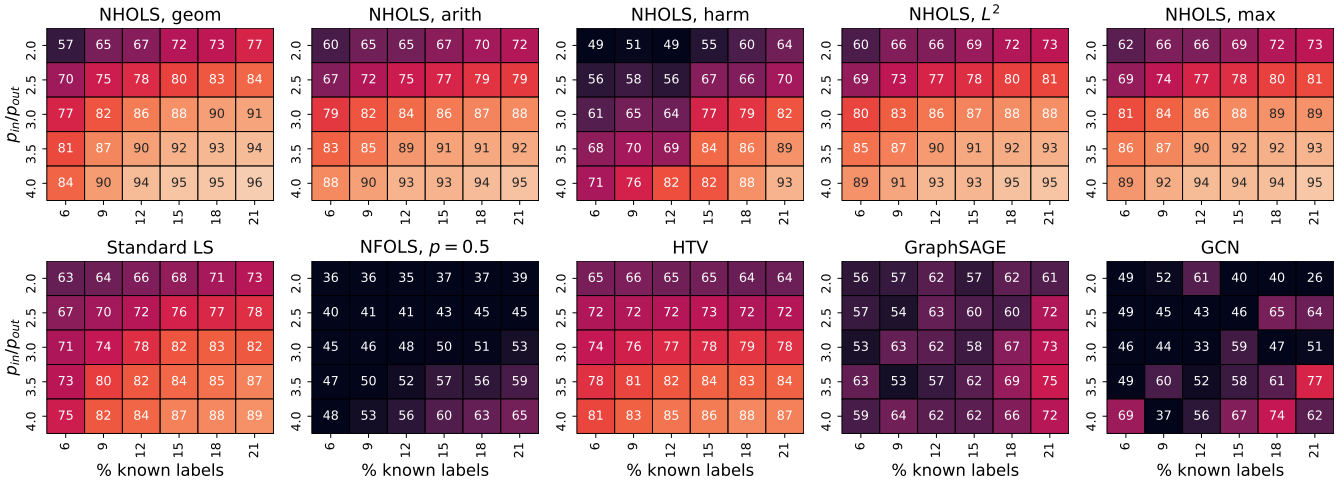

Figure 1: Accuracy on synthetic stochastic block models. Each table corresponds to different method, and table entries are the mean accuracy over 10 random instances with the given parameter settings. Overall, the various NHOLS methods perform much better than the baselines.

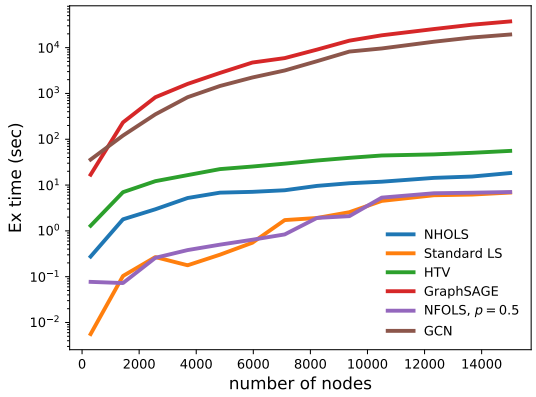

Figure 2: Running times for one hyperparameter setting with SBM data.

of the principal component embeddings, but both resulted in much worse performance.

We recorded the mean accuracy and standard deviation of five random samples each of various amounts of labeled nodes (Table 2). We find that incorporating higher-order information with some mixing function is more accurate than LS in nearly all cases and is also more accurate than the other baselines. The absolute accuracy of these methods is also quite remarkable. For instance, the L^2 mixing function achieves 91.6% mean accuracy with just 0.1% of MNIST points labeled (6 labeled points per class), and the maximum mixing function achieves 92.1% mean accuracy with just 0.4% of pendigits points labeled (5 labeled points per class). Also, a single mixing function tends to have the best performance for a fixed dataset, regardless of the number of labels (e.g., the maximum mixing function for pendigits or the L^2 mixing function for MNIST).

Finally, the GNN model performance is often poor, even in cases where meaningful node features are available. This is likely a result of having only a small percentage of labeled examples. For example, it is common to have over 15% of the nodes labeled just as a validation set for hyperparameter tuning [32]. Still, even with 20% of nodes labeled and hyperparameter tuning, both GNNs perform substantially worse than standard label spreading or NHOLS on the Facebook graphs. The simpler embedding method Planetoid tends to be better than the GNNs. We also experimented with more sophisticated GNN architectures, but they overfit and were less accurate.

5 DISCUSSION

We have developed a natural and substantial extension of traditional label spreading for semi-supervised learning that can incorporate a broad range of nonlinearities into the spreading process. These nonlinearities come from mixing functions that operate on higher-order information associated with the data. Given the challenges in developing optimization results of nonlinear functions, it is remarkable that we can achieve a sound theoretical framework for such an expressive spreading process. We provided guarantees on convergence of the iterations to a unique solution, and we showed that the process optimizes a meaningful objective function.

For the datasets that we considered, the performance of label spreading methods was far superior to "state-of-the-art" graph neural network approaches, in terms of both runtime and prediction accuracy. Part of the reason is that we were in a regime with a small number of labels, which is an atypical setting for neural network based methods on graphs. However, even with 20% of nodes labeled on social network data, embedding-based approaches still had subpar performance. The strong smoothness assumptions of the label spreading model — at both the first-order graph level and the higher-order hypergraph level — were key to good performance.

The convergence result in Theorem 3.1 is much more general than is needed for the NHOLS method. This provides a new way to analyze nonlinearities in graph-based methods that we expect

Table 2: Mean prediction accuracy (plus or minus one standard deviation) over five random samples of labeled nodes, six datasets, and four percentages of labeled nodes. We compare our NHOLS method using five different mixing functions to standard LS, nonlinear first-order label spreading (NFOLS, $p = 0.5$ in (17)), hypergraph total variation minimization (HTV) [27], two Graph Neural Network models [25, 32], and Planetoid [59]. Incorporating higher-order information into label spreading with NHOLS leads to more accurate predictions compared to the baselines.

		Rice31 (n = 3560)				Caltech36 (n = 590)			
method	% labeled	5.0%	10.0%	15.0%	20.0%	5.0%	10.0%	15.0%	20.0%
NHOLS, arith		87.9 ±0.5	**90.2** ±0.7	90.8 ±0.4	91.2 ±0.4	82.1 ±1.6	83.9 ±0.6	83.8 ±1.4	84.2 ±2.0
NHOLS, harm		87.1 ±1.0	**90.2** ±0.8	**91.0** ±0.5	**91.4** ±0.4	77.0 ±2.3	82.0 ±0.8	83.3 ±1.9	83.9 ±2.3
NHOLS, L^2		87.8 ±0.8	**90.2** ±0.7	90.7 ±0.3	**91.4** ±0.2	**82.2** ±0.9	**84.0** ±0.9	83.5 ±1.4	**84.4** ±1.6
NHOLS, geom		87.4 ±0.7	90.1 ±0.6	90.9 ±0.3	**91.4** ±0.4	79.6 ±1.1	83.0 ±0.5	83.9 ±1.3	84.0 ±2.3
NHOLS, max		**88.1** ±0.6	**90.2** ±0.7	90.8 ±0.3	90.9 ±0.1	82.1 ±1.0	83.6 ±1.5	**84.0** ±1.3	84.0 ±1.6
Standard LS		82.6 ±0.9	87.8 ±0.8	89.4 ±0.3	90.6 ±0.5	74.4 ±1.5	77.8 ±1.2	78.3 ±2.7	80.1 ±1.9
NFOLS (p = 0.5)		81.5 ±1.2	87.1 ±1.6	88.8 ±0.6	90.1 ±0.7	68.6 ±4.9	75.1 ±0.9	76.5 ±0.5	76.5 ±2.1
HTV		81.6 ±0.1	85.7 ±0.7	87.7 ±0.7	90.0 ±0.0	66.1 ±0.3	76.1 ±1.1	75.9 ±0.4	81.8 ±0.7
GCN		80.5 ±2.7	81.2 ±2.9	83.4 ±3.3	85.7 ±3.0	55.3 ±11.3	60.0 ±8.9	69.6 ±7.3	67.2 ±6.9
GraphSAGE		64.2 ±4.4	76.0 ±2.8	79.5 ±2.0	83.1 ±1.0	54.5 ±5.8	64.7 ±4.6	69.3 ±6.3	72.0 ±2.0
Planetoid		71.9 ±1.1	83.1 ±1.5	86.4 ±0.9	87.2 ±0.8	74.4 ±2.1	78.3 ±0.8	78.4 ±1.2	80.3 ±1.3
		optdigits (n = 5620)				pendigits (n = 10992)			
method	% labeled	0.4%	0.7%	1.0%	1.3%	0.4%	0.7%	1.0%	1.3%
NHOLS, arith		93.6 ±1.6	95.2 ±1.5	**96.6** ±1.2	**96.7** ±1.3	89.2 ±2.4	94.2 ±0.9	94.9 ±0.8	**96.7** ±1.2
NHOLS, harm		93.2 ±1.9	94.7 ±1.5	96.2 ±1.4	96.4 ±1.4	86.1 ±3.7	92.6 ±1.3	93.6 ±1.0	95.7 ±1.4
NHOLS, L^2		**93.7** ±1.5	**95.6** ±1.2	96.5 ±1.3	**96.7** ±1.3	89.2 ±2.7	94.3 ±1.0	95.2 ±0.6	96.6 ±1.3
NHOLS, geom		93.3 ±2.1	95.0 ±1.6	96.2 ±1.4	96.4 ±1.5	86.5 ±2.8	93.0 ±1.3	94.0 ±1.0	95.9 ±1.5
NHOLS, max		**93.7** ±2.9	**95.6** ±1.1	96.2 ±1.6	96.5 ±1.3	**92.1** ±1.9	**95.1** ±0.7	**95.9** ±0.5	96.6 ±1.1
Standard LS		91.2 ±4.2	94.7 ±1.3	95.3 ±1.6	95.5 ±1.8	89.3 ±2.1	94.0 ±1.0	94.6 ±0.5	96.0 ±1.1
NFOLS (p = 0.5)		65.9 ±8.0	73.6 ±5.4	82.3 ±3.7	87.8 ±1.8	55.7 ±7.0	62.5 ±3.0	67.1 ±1.3	76.5 ±2.7
HTV		87.0 ±5.2	90.9 ±2.6	93.3 ±0.7	94.1 ±1.5	82.2 ±2.1	91.3 ±1.8	93.3 ±0.5	93.3 ±0.9
GCN		63.4 ±8.5	69.3 ±3.5	76.9 ±1.7	82.4 ±2.4	62.9 ±2.0	71.4 ±3.8	78.0 ±3.4	80.5 ±4.3
GraphSAGE		55.9 ±2.5	57.9 ±6.1	67.3 ±3.8	71.0 ±5.7	49.3 ±15.8	56.0 ±17.5	61.2 ±6.8	64.7 ±10.8
Planetoid		74.3 ±4.1	79.6 ±2.7	81.9 ±2.1	85.0 ±2.2	75.5 ±2.1	80.4 ±1.3	80.8 ±2.3	84.8 ±1.1
		MNIST (n = 60000)				Fashion-MNIST (n = 60000)			
method	% labeled	0.1%	0.3%	0.5%	0.7%	0.1%	0.3%	0.5%	0.7%
NHOLS, arith		91.0 ±1.0	95.1 ±0.3	95.7 ±0.3	95.8 ±0.1	70.9 ±2.7	75.2 ±1.0	**78.0** ±0.6	**78.7** ±0.7
NHOLS, harm		89.2 ±1.6	94.5 ±0.4	95.1 ±0.4	95.3 ±0.2	70.4 ±1.6	74.9 ±1.1	77.4 ±0.6	78.4 ±0.4
NHOLS, L^2		**91.6** ±0.7	**95.3** ±0.3	**95.8** ±0.3	**95.9** ±0.1	**72.0** ±1.5	**75.3** ±1.1	77.7 ±1.0	**78.7** ±0.6
NHOLS, geom		89.5 ±1.4	94.6 ±0.3	95.2 ±0.4	95.4 ±0.2	70.7 ±1.5	74.8 ±1.2	77.5 ±0.7	78.5 ±0.5
NHOLS, max		91.0 ±1.4	95.0 ±0.5	95.5 ±0.5	95.8 ±0.1	71.6 ±1.7	74.4 ±1.6	77.4 ±0.8	78.4 ±0.9
Standard LS		86.8 ±1.2	92.7 ±0.5	93.5 ±0.5	94.1 ±0.3	70.4 ±1.0	74.3 ±0.9	76.7 ±0.8	78.1 ±0.5
NFOLS (p = 0.5)		40.1 ±2.7	50.0 ±1.4	57.0 ±2.1	56.1 ±1.4	34.8 ±2.8	51.4 ±2.4	59.6 ±3.4	67.0 ±2.3
HTV		79.7 ±3.4	88.3 ±0.9	90.1 ±0.8	90.7 ±0.4	59.8 ±2.3	68.6 ±1.9	70.2 ±0.4	72.0 ±0.6
GCN		66.5 ±1.8	80.6 ±0.9	85.5 ±0.8	85.7 ±1.4	65.5 ±2.6	72.9 ±1.8	75.0 ±0.8	76.5 ±0.8
GraphSAGE		59.2 ±2.1	75.5 ±1.0	81.1 ±2.0	82.6 ±1.0	59.9 ±1.4	69.0 ±2.1	72.1 ±1.4	72.2 ±1.3
Planetoid		68.4 ±2.7	79.1 ±0.8	82.3 ±0.7	83.7 ±0.9	68.0 ±2.1	73.3 ±0.8	75.9 ±0.8	78.1 ±0.6

will be useful for future research. For example, other nonlinear iterations could easily be developed, but there is still a challenge in finding corresponding interpretable objective functions that are optimized by the iterations, as we were able to do for NHOLS.

Acknowledgments.

We thank Matthias Hein for supplying the code that implements the HTV algorithm [27]. We thank David Gleich and Antoine Gautier for helpful discussions. ARB is supported in party by NSF Award DMS-1830274, ARO Award W911NF19-1-0057, ARO MURI, and JPMorgan Chase & Co. FT is partially supported by INdAM-GNCS.

REFERENCES

[1] Sameer Agarwal, Kristin Branson, and Serge Belongie. 2006. Higher order learning with graphs. In *Proceedings of the 23rd international conference on Machine learning*. 17–24.

[2] Nesreen K Ahmed, Jennifer Neville, Ryan A Rossi, Nick G Duffield, and Theodore L Willke. 2017. Graphlet decomposition: Framework, algorithms, and applications. *Knowledge and Information Systems* 50, 3 (2017), 689–722.

[3] Morteza Alamgir and Ulrike V Luxburg. 2011. Phase transition in the family of p-resistances. In *Advances in Neural Information Processing Systems*. 379–387.

[4] Fevzi Alimoglu and Ethem Alpaydin. 1996. Methods of combining multiple classifiers based on different representations for pen-based handwritten digit recognition. In *Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium*. Citeseer.

[5] Francesca Arrigo, Desmond J Higham, and Francesco Tudisco. 2020. A framework for second-order eigenvector centralities and clustering coefficients. *Proceedings of the Royal Society A* 476, 2236 (2020), 20190724.

[6] Francesca Arrigo and Francesco Tudisco. 2019. Multi-dimensional, multilayer, nonlinear and dynamic HITS. In *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 369–377.

[7] Austin R Benson. 2019. Three hypergraph eigenvector centralities. *SIAM Journal on Mathematics of Data Science* 1, 2 (2019), 293–312.

[8] Austin R Benson, Rediet Abebe, Michael T Schaub, Ali Jadbabaie, and Jon Kleinberg. 2018. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences* 115, 48 (2018), E11221–E11230.

[9] Austin R Benson, David F Gleich, and Jure Leskovec. 2016. Higher-order organization of complex networks. *Science* 353, 6295 (2016), 163–166.

[10] Jonathan W Berry, Luke K Fostvedt, Daniel J Nordman, Cynthia A Phillips, C Seshadhri, and Alyson G Wilson. 2014. Why do simple algorithms for triangle enumeration work in the real world?. In *Proceedings of the 5th conference on Innovations in Theoretical Computer Science*. 225–234.

[11] Nick Bridle and Xiaojin Zhu. 2013. p-voltages: Laplacian regularization for semi-supervised learning on high-dimensional data. In *Eleventh Workshop on Mining and Learning with Graphs (MLG2013)*. Citeseer.

[12] T-H Hubert Chan, Anand Louis, Zhihao Gavin Tang, and Chenzi Zhang. 2018. Spectral properties of hypergraph laplacian and approximation algorithms. *Journal of the ACM (JACM)* 65, 3 (2018), 1–48.

[13] Sudhanshu Chanpuriya and Cameron Musco. 2020. InfiniteWalk: Deep Network Embeddings as Laplacian Embeddings with a Nonlinearity. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[14] Sudhanshu Chanpuriya, Cameron Musco, Konstantinos Sotiropoulos, and Charalampos E Tsourakakis. 2020. Node Embeddings and Exact Low-Rank Representations of Complex Networks. In *Advances in Neural Information Processing Systems*.

[15] Alex Chin, Yatong Chen, Kristen M. Altenburger, and Johan Ugander. 2019. Decoupled smoothing on graphs. In *The World Wide Web Conference*. 263–272.

[16] Uthsav Chitra and Benjamin Raphael. 2019. Random Walks on Hypergraphs with Edge-Dependent Vertex Weights. In *International Conference on Machine Learning*. 1172–1181.

[17] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[18] Dhivya Eswaran, Srijan Kumar, and Christos Faloutsos. 2020. Higher-Order Label Homogeneity and Spreading in Graphs. In *Proceedings of The Web Conference 2020*. 2493–2499.

[19] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3558–3565.

[20] Yasuhiro Fujiwara and Go Irie. 2014. Efficient label propagation. In *International Conference on Machine Learning*. 784–792.

[21] Antoine Gautier and Francesco Tudisco. 2019. The contractivity of cone-preserving multilinear mappings. *Nonlinearity* 32 (2019), 4713.

[22] Antoine Gautier, Francesco Tudisco, and Matthias Hein. 2019. The Perron-Frobenius theorem for multihomogeneous mappings. *SIAM J. Matrix Anal. Appl.* 40, 3 (2019), 1179–1205.

[23] David F Gleich and Michael W Mahoney. 2015. 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*. 359–368.

[24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[25] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.

[26] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin* (2017).

[27] Matthias Hein, Simon Setzer, Leonardo Jost, and Syama Sundar Rangapuram. 2013. The total variation on hypergraphs - Learning on hypergraphs revisited. In *Advances in Neural Information Processing Systems*. 2427–2435.

[28] Rania Ibrahim and David Gleich. 2019. Nonlinear Diffusion for Community Detection and Semi-Supervised Learning. In *The World Wide Web Conference*. 739–750.

[29] Rania Ibrahim and David F Gleich. 2020. Local Hypergraph Clustering using Capacity Releasing Diffusion. *arXiv:2003.04213* (2020).

[30] Thorsten Joachims. 2003. Transductive learning via spectral graph partitioning. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 290–297.

[31] Varun Kanade, Elchanan Mossel, and Tselil Schramm. 2016. Global and local information in clustering labeled block models. *IEEE Transactions on Information Theory* 62, 10 (2016), 5906–5917.

[32] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[33] Isabel M Kloumann, Johan Ugander, and Jon Kleinberg. 2017. Block models and personalized PageRank. *Proceedings of the National Academy of Sciences* 114, 1 (2017), 33–38.

[34] Rasmus Kyng, Anup Rao, Sushant Sachdeva, and Daniel A Spielman. 2015. Algorithms for Lipschitz learning on graphs. In *Conference on Learning Theory*. 1190–1223.

[35] Matthieu Latapy. 2008. Main-memory triangle computations for very large (sparse (power-law)) graphs. *Theoretical computer science* 407, 1-3 (2008), 458–473.

[36] Yann LeCun, Corinna Cortes, and CJ Burges. 2010. MNIST handwritten digit database.

[37] Bas Lemmens and Roger Nussbaum. 2012. *Nonlinear Perron-Frobenius Theory*. Vol. 189. Cambridge University Press.

[38] Pan Li, I Chien, and Olgica Milenkovic. 2019. Optimizing Generalized PageRank Methods for Seed-Expansion Community Detection. In *Advances in Neural Information Processing Systems*. 11705–11716.

[39] Pan Li, Niao He, and Olgica Milenkovic. 2020. Quadratic decomposable submodular function minimization: Theory and practice. *Journal of Machine Learning Research* 21, 106 (2020), 1–49.

[40] Pan Li and Olgica Milenkovic. 2017. Inhomogeneous hypergraph clustering with applications. In *Advances in Neural Information Processing Systems*. 2308–2318.

[41] Pan Li and Olgica Milenkovic. 2018. Submodular Hypergraphs: p-Laplacians, Cheeger Inequalities and Spectral Clustering. In *International Conference on Machine Learning*. 3014–3023.

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

[43] Pedro Mercado, Francesco Tudisco, and Matthias Hein. 2019. Generalized Matrix Means for Semi-Supervised Learning with Multilayer Graphs. In *Advances in Neural Information Processing Systems*. 14848–14857.

[44] Elchanan Mossel and Jiaming Xu. 2016. Local algorithms for block models with side information. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*. 71–80.

[45] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When does label smoothing help?. In *Advances in Neural Information Processing Systems*. 4696–4705.

[46] Huda Nassar, Caitlin Kennedy, Shweta Jain, Austin R Benson, and David Gleich. 2020. Using Cliques with Higher-order Spectral Embeddings Improves Graph Visualizations. In *Proceedings of The Web Conference 2020*. 2927–2933.

[47] Ryan A Rossi, Nesreen K Ahmed, and Eunyee Koh. 2018. Higher-order network representation learning. In *Companion Proceedings of the The Web Conference 2018*. 3–4.

[48] Ryan A Rossi, Anup Rao, Sungchul Kim, Eunyee Koh, Nesreen K Ahmed, and Gang Wu. 2019. Higher-order ranking and link prediction: From closing triangles to closing higher-order motifs. *arXiv preprint arXiv:1906.05059* (2019).

[49] Sai Nageswar Satchidanand, Harini Ananthapadmanaban, and Balaraman Ravindran. 2015. Extended discriminative random walk: a hypergraph approach to multi-view multi-relational transductive learning. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[50] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2818–2826.

[51] Amanda L Traud, Peter J Mucha, and Mason A Porter. 2012. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications* 391, 16 (2012), 4165–4180.

[52] Charalampos E Tsourakakis, Jakub Pachocki, and Michael Mitzenmacher. 2017. Scalable motif-aware graph clustering. In *Proceedings of the 26th International Conference on World Wide Web*. 1451–1460.

[53] Nate Veldt, Austin R Benson, and Jon Kleinberg. 2020. Hypergraph Cuts with General Splitting Functions. *arXiv preprint arXiv:2001.02817* (2020).

[54] Nate Veldt, Austin R Benson, and Jon Kleinberg. 2020. Minimizing Localized Ratio Cut Objectives in Hypergraphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1708–1718.

[55] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).
[56] Lei Xu, Adam Krzyzak, and Ching Y Suen. 1992. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE transactions on systems, man, and cybernetics* 22, 3 (1992), 418–435.
[57] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. 2019. HyperGCN: A New Method For Training Graph Convolutional Networks on Hypergraphs. In *Advances in Neural Information Processing Systems*. 1509–1520.
[58] Shenghao Yang, Di Wang, and Kimon Fountoulakis. 2020. p-Norm Flow Diffusion for Local Graph Clustering. *arXiv:2005.09810* (2020).
[59] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*. PMLR, 40–48.
[60] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. 2017. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 555–564.
[61] Muhan Zhang, Zhicheng Cui, Shali Jiang, and Yixin Chen. 2018. Beyond link prediction: Predicting hyperlinks in adjacency space. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
[62] Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. In *Advances in neural information processing systems*. 321–328.
[63] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2007. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in neural information processing systems*. 1601–1608.
[64] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2018. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434* (2018).
[65] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*. 912–919.
[66] Xiaojin Zhu and Andrew B Goldberg. 2009. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning* 3, 1 (2009), 1–130.

A PROOF OF THEOREM 3.1

We start with the proof of Theorem 3.1. Let $\varrho\colon \mathbb{R}^n \to \mathbb{R}$ be positive, order-preserving, and 1-homogeneous and let $F\colon \mathbb{R}^n \to \mathbb{R}^n$ be p-homogeneous, order-preserving and positive. Let $\mathbb{R}^n_+$ and $\mathbb{R}^n_{++}$ denote the set of entry-wise nonnegative and entry-wise positive vectors in $\mathbb{R}^n$, respectively. Also, for a set of vectors Σ, let Σ/ϱ denote the slice $\Sigma/\varrho = \{f \in \Sigma : \varrho(f) = 1\}$. We first need the following lemma.

Lemma A.1. *Let $a, b, c > 0$. Then*

$$\log\Big(\frac{a+c}{b+c}\Big) \le \frac{a}{a+c}\log\Big(\frac{a}{b}\Big)$$

Proof. Let $g(x) = x\log(x)$. We have

$$\frac{a+c}{b+c}\log\Big(\frac{a+c}{b+c}\Big) = g\Big(\frac{a+c}{b+c}\Big) = g\Big(\frac{b}{b+c}\,\frac{a}{b} + \frac{c}{b+c}\Big).$$

As $\frac{b}{b+c} + \frac{c}{b+c} = 1$ and g is convex, we can apply Jensen's inequality to get

$$g\Big(\frac{b}{b+c}\,\frac{a}{b} + \frac{c}{b+c}\Big) \le \frac{b}{b+c}\,g\Big(\frac{a}{b}\Big) + \frac{c}{b+c}\,g(1) = \frac{a}{b+c}\log\Big(\frac{a}{b}\Big).$$

Combining all together we get

$$\frac{a+c}{b+c}\log\Big(\frac{a+c}{b+c}\Big) \le \frac{a}{b+c}\log\Big(\frac{a}{b}\Big)$$

which yields the claims. □

Now we give the proof of Theorem 3.1.

Proof. This result falls within the family of contraction-type theorems for nonlinear mappings on abstract cones [37]. Here we provide a simple and self-contained proof. Define

$$G(f) = F(f) + y \qquad \text{and} \qquad \widetilde{G}(f) = \frac{G(f)}{\varrho(G(f))}.$$

Notice that $f^{(r+1)} = \widetilde{G}(f^{(r)})$, for all $r = 0, 1, 2, \dots$ and that, by assumption, $G(f) > 0$ for all $f > 0$. Thus $\varrho(G(f)) > 0$ and $\widetilde{G}(f)$ is well defined on $\mathbb{R}^n_{++}$. Moreover, note that by the assumption $F(f) \le Cy$ we have $y \le G(f) \le (C+1)y$. Therefore, letting $C_1 = \varrho(y)$ and since ϱ is order-preserving, we have

$$\delta_1 y := \frac{1}{(C+1)C_1}y \le \widetilde{G}(f) = \frac{G(f)}{\varrho(G(f))} \le \frac{C+1}{C_1}y =: \delta_2 y \tag{19}$$

for all $f \in \mathbb{R}^n_{++}$. Consider the set

$$\Sigma = \big\{f \in \mathbb{R}^n_{++} : \delta_1 y \le f \le \delta_2 y\big\} \subseteq \mathbb{R}^n_{++}.$$

By (19), we have that $\widetilde{G}$ preserves the slice $\Sigma/\varrho \subseteq \mathbb{R}^n_{++}/\varrho$, that is $\widetilde{G}(f) \in \Sigma/\varrho$ for all $f \in \Sigma/\varrho$.

For two points $u, v \in \mathbb{R}^n_{++}$, consider the Hilbert distance

$$d(u, v) = \log\Big(\frac{M(u/v)}{m(u/v)}\Big),$$

where $M(u/v) = \max_i u_i/v_i$ and $m(u/v) = \min_i u_i/v_i$. With this definition we can equivalently write

$$\Sigma = \big\{f \in \mathbb{R}^n_{++} : d(f, y) \le \delta_2/\delta_1\big\},$$

thus, as ϱ is 1-homogeneous, Σ/ϱ equipped with the Hilbert distance is a compact metric space [37]. In order to conclude the proof, it is sufficient to show that $\widetilde{G}$ is a contraction with respect to the Hilbert metric. In fact, the sequence $f^{(r)}$ belongs to Σ/ϱ for any $f^{(0)}$ and since $(\Sigma/\varrho, d)$ is compact, the sequence $f^{(r)}$ converges to the unique fixed point f^* of $\widetilde{G}$ in Σ/ϱ.

We show below that $\widetilde{G}$ is a contraction. To this end, first note that by definition we have $d(\widetilde{G}(u), \widetilde{G}(v)) = d(G(u), G(v))$. Now note that, as F is p-homogeneous and order-preserving with $p \in [0, 1]$ and ϱ is 1-homogeneous and oder-preserving, for any $u, v \in \mathbb{R}^n_{++}$ we have

$$m(u/v)^p F(v) = F(m(u/v)v) \le F(u) \tag{20}$$
$$M(u/v)^p F(v) = F(M(u/v)v) \ge F(u) \tag{21}$$
$$m(u/v)\varrho(v) = \varrho(m(u/v)v) \le \varrho(u) \tag{22}$$
$$M(u/v)\varrho(v) = \varrho(M(u/v)v) \ge \varrho(u) \tag{23}$$

Moreover, for any $u, v \in \mathbb{R}^n_{++}/\varrho$ it holds that

$$m(u/v) = \varrho(v)m(u/v) \le \varrho(v) = 1 = \varrho(u) \le \varrho(v)M(u/v) = M(u/v),$$

so

$$m(u/v) \le m(u/v)^p \le 1 \le M(u/v)^p \le M(u/v). \tag{24}$$

By assumption, there exists $C > 0$ such that $F(u) \le Cy$, for all $u \in \mathbb{R}^n_{++}/\varrho$. Therefore, using (20)–(24), for any $u, v \in \mathbb{R}^n_{++}/\varrho$ we

have

$$\begin{aligned}
&(m(u/v)C+1)(F(v)+y)\\
&=(m(u/v)C+m(u/v)-m(u/v)+1)F(v)+(m(u/v)C+1)y\\
&=(C+1)m(u/v)F(v)+(1-m(u/v))F(v)+(m(u/v)C+1)y\\
&\le (C+1)m(u/v)^p F(v)+(1-m(u/v))F(v)+(m(u/v)C+1)y\\
&\le (C+1)F(u)+(1-m(u/v))Cy+(m(u/v)c+1)y\\
&=(C+1)(F(u)+y),
\end{aligned}$$

where we used the fact that $(1-m(u/v))\ge 0$ to get the inequality $(1-m(u/v))F(v)\le(1-m(u/v))Cy$. Thus,

$$m(G(u)/G(v))=m\big((F(u)+y)/(F(v)+y)\big)\ge (m(u/v)C+1)/(C+1)\,.$$

Similarly, as $(1-M(u/v))\le 0$, we have

$$\begin{aligned}
&(M(u/v)C+1)(F(v)+y)\\
&\ge CF(u)+F(u)+(1-M(u/v))Cy+(M(u/v)C+1)y\\
&=(C+1)(F(u)+y),
\end{aligned}$$

which gives $M(G(u)/G(v))\le (M(u/v)C+1)/(C+1)$. Therefore,

$$\begin{aligned}
d(G(u),G(v))&=\log\Big(\frac{M(G(u)/G(v))}{m(G(u)/G(v))}\Big)\\
&\le \log\Big(\frac{M(u/v)C+1}{m(u/v)C+1}\Big)=\log\Big(\frac{M(u/v)+\delta}{m(u/v)+\delta}\Big)
\end{aligned}$$

with $\delta=1/C$. Finally, using Lemma A.1, we get

$$d(\widetilde{G}(u),\widetilde{G}(v))=d(G(u),G(v))\le\left(\frac{M(u/v)}{M(u/v)+\delta}\right)d(u,v)<d(u,v),$$

which shows that $\widetilde{G}$ is a contraction in the compact metric space $(\Sigma/\varrho,d)$, concluding the proof. □

B PROOF OF COROLLARY 3.2

Consider now the case where

$$\varrho(f)=\varphi(f)=\frac{1}{2}\sqrt{\sum_{ij}B_{ij}\,\sigma\Big(\frac{f_i}{\sqrt{\delta_i}},\frac{f_j}{\sqrt{\delta_j}}\Big)^2}\,.$$

We start with a technical lemma.

Lemma B.1. *Assume that σ is 1-homogeneous and positive and that both y and $\alpha D_H^{-1/2}\mathcal{A}\sigma(D_H^{-1/2}f)+\beta D_G^{-1/2}AD_G^{-1/2}f$ are entry-wise positive, for every $f\in\mathbb{R}^n_{++}$. Then, there exists $C>0$ such that*

$$\alpha D_H^{-1/2}\mathcal{A}\sigma(D_H^{-1/2}f)+\beta D_G^{-1/2}AD_G^{-1/2}f\le Cy$$

for all $f\in\mathbb{R}^n_{++}/\varphi$.

Proof. Since we are assuming that every node has hyper-degree $\delta_i=\sum_{jk}\mathcal{A}_{ijk}>0$, for every i there exist j and k such that ijk is a hyperedge. Thus, if $U\subseteq V\times V$ is the set of nonzero entries of the matrix

$$B=(B_{ij})=\Big(\sum\nolimits_k\mathcal{A}_{kij}\Big)\,,$$

then the pairs in U must contain all the nodes, i.e. for all $k\in V$ there exists $(i,j)\in U$ such that $i=k$ or $j=k$. Therefore, if $f>0$ and $\varphi(f)=1$ then f must be entry-wise bounded. Hence, as φ is positive and 1-homogeneous, we can choose

$$C=\frac{1}{\min_i y_i}\max_i\max_{f\in\mathbb{R}^n_{++}}\frac{(\alpha D_H^{-1/2}\mathcal{A}\sigma(D_H^{-1/2}f)+\beta D_G^{-1/2}AD_G^{-1/2}f)_i}{\varphi(f)}$$

to obtain the claim. □

Now we prove Corollary 3.2.

Proof. Let $F(f)=\alpha D_H^{-1/2}\mathcal{A}\sigma(D_H^{-1/2}f)+\beta D_G^{-1/2}AD_G^{-1/2}f$ and $y=\gamma y_\varepsilon$. By Lemma B.1 all the assumptions of Theorem 3.1 are satisfied, and the convergence follows. □

C PROOF OF THEOREM 3.3

We again start with a useful lemma.

Lemma C.1. *Let $E\colon\mathbb{R}^n\to\mathbb{R}_+$ be defined by*

$$E(f)=\frac{f^\top(D_Hf-F(f))}{2}\tag{25}$$

with $F\colon\mathbb{R}^n\to\mathbb{R}^n$ differentiable and such that $F(f)\ge 0$ for all $f\ge 0$ and $F(\alpha f)=\alpha F(f)$ for all $\alpha\ge 0$. Then $\nabla E(f)=D_Hf-F(f)$.

Proof. This is a relatively direct consequence of Euler's theorem for homogeneous functions. For completeness, we provide a self-contained proof here. Consider the function $G(\alpha)=F(\alpha f)-\alpha F(f)$. Then G is differentiable and $G(\alpha)=0$ for all $\alpha\ge 0$. Thus, $G'(\alpha)=0$ for all α in a neighborhood of $\alpha_0=1$. For any such α, we have

$$G'(\alpha)=\alpha JF(\alpha f)f-F(f)=0,$$

where $JF(f)$ denotes the Jacobian of F evaluated at f. Evaluating G' on $\alpha=1$ we get $JF(f)f=F(f)$. Therefore,

$$\begin{aligned}
2\nabla E(f)&=\nabla\{f^\top(D_Hf-F(f))\}\\
&=D_Hf-F(f)+\Big(D_H-JF(f)\Big)f=2D_Hf-2F(f),
\end{aligned}$$

which gives us the claim. □

Now we prove Theorem 3.3.

Proof. Let

$$E_1(f)=\sum_{ij}A_{ij}\Big(\frac{f_i}{\sqrt{d_i}}-\frac{f_j}{\sqrt{d_j}}\Big)^2$$

and

$$E_2(f)=\sum_{ijk}\mathcal{A}_{ijk}\Big(\frac{f_i}{\sqrt{\delta_i}}-\frac{1}{2}\sigma\Big(\frac{f_j}{\sqrt{\delta_j}},\frac{f_k}{\sqrt{\delta_k}}\Big)\Big)^2,$$

and consider the following modified loss

$$\widetilde{\vartheta}(f)=\frac{1}{2}\left\{\left\|f-\frac{y}{\varphi(y)}\right\|^2+\lambda E_1(f)+\mu E_2(f)-\mu\varphi(f)^2\right\}.$$

Subject to the constraint $\varphi(f)=1$, the minimizing points of $\widetilde{\vartheta}$ and those of ϑ in (13) coincide. We show that the gradient of the loss function $\widetilde{\vartheta}$ vanishes on $f^*>0$ with $\varphi(f^*)=1$ if and only if f^* is a fixed point for the iterator of Algorithm 1.

For simplicity, let us write $\widetilde{y}=y/\varphi(y)$ with $y>0$. We have $\nabla\|f-\widetilde{y}\|^2=2(f-\widetilde{y})$ and $\nabla E_1(f)=2\Delta f=2(I-D_G^{-1}AD_G^{-1})f$. As for E_2, observe that from $\sum_{jk}\mathcal{A}_{ijk}=\delta_i$,we get

$$\begin{aligned}
f^\top(D_Hf-\mathcal{A}\sigma(f))&=\sum_i\delta_if_i^2-\sum_{ijk}f_i\mathcal{A}_{ijk}\sigma(f_j,f_k)\\
&=\sum_{ijk}\mathcal{A}_{ijk}\big(f_i^2-f_i\sigma(f_j,f_k)\big)\\
&=\sum_{ijk}\mathcal{A}_{ijk}\Big(f_i-\frac{\sigma(f_j,f_k)}{2}\Big)^2-\frac{1}{4}\sum_{jk}B_{jk}\sigma(f_j,f_k)^2
\end{aligned}$$

with $B_{jk} = \sum_i \mathcal{A}_{ijk}$. Thus,

$$f^\top(D_H f - \mathcal{A}\sigma(f)) = E_2(D_H^{1/2} f) - \varphi(D_H^{1/2} f)^2 .$$

Now, since $\sigma(f)$ is 1-homogeneous and differentiable, so is $F(f) = \mathcal{A}\sigma(f)$, and using Lemma C.1 we obtain

$$\nabla\{E_2(D_H^{1/2} f) - \varphi(D_H^{1/2} f)^2\} = 2(D_H f - \mathcal{A}\sigma(f)),$$

which, with the change of variable $f \mapsto D_H^{-1/2} f$, yields

$$\nabla\{E_2(f) - \varphi(f)^2\} = 2\big(f - D_H^{-1/2}\mathcal{A}\sigma(D_H^{-1/2} f)\big).$$

Altogether, we have that

$$\begin{aligned}
&\nabla\widetilde{\vartheta}(f) = \\
&= f - \widetilde{y} + \lambda(I - D_G^{-1/2} A D_G^{-1/2}) f + \mu\{f - D_H^{-1/2}\mathcal{A}\sigma(D_H^{-1/2} f)\} \\
&= (1 + \lambda + \mu) f - \lambda D_G^{-1/2} A D_G^{-1/2} f - \mu D_H^{-1/2}\mathcal{A}\sigma(D_H^{-1/2} f) - \widetilde{y},
\end{aligned}$$

which implies that $f^* \in \mathbb{R}^n_{++}/\varphi$ is such that $\nabla\widetilde{\vartheta}(f^*) = 0$ if and only if f^* is a fixed point of NHOLS, i.e.,

$$f^* = \alpha D_H^{-1/2}\mathcal{A}\sigma(D_H^{-1/2} f^*) + \beta D_G^{-1/2} A D_G^{-1/2} f^* + \gamma\widetilde{y}$$

with $\lambda = \beta/\gamma$, $\mu = \alpha/\gamma$ and $\alpha + \beta + \gamma = 1$.

Finally, by Corollary 3.2 we know that the NHOLS iterations in Algorithm 1 converge to $f^* \in \mathbb{R}^n_{++}/\varphi$ for all positive starting points. Moreover, f^* is the unique fixed point in the slice $\mathbb{R}^n_{++}/\varphi$. As ϑ and $\widetilde{\vartheta}$ have the same minimizing points on that slice, this shows that f^* is the global solution of $\min\{\vartheta(f) : f \in \mathbb{R}^n_{++}/\varphi\}$, concluding the proof.

□