Elastic Graph Neural Networks

Xiaorui Liu [1 *] **Wei Jin** [1 *] **Yao Ma** [1] **Yaxin Li** [1] **Hua Liu** [2] **Yiqi Wang** [1] **Ming Yan** [3] **Jiliang Tang** [1]

Abstract

While many existing graph neural networks (GNNs) have been proven to perform ℓ_2-based graph smoothing that enforces smoothness globally, in this work we aim to further enhance the local smoothness adaptivity of GNNs via ℓ_1-based graph smoothing. As a result, we introduce a family of GNNs (Elastic GNNs) based on ℓ_1 and ℓ_2-based graph smoothing. In particular, we propose a novel and general message passing scheme into GNNs. This message passing algorithm is not only friendly to back-propagation training but also achieves the desired smoothing properties with a theoretical convergence guarantee. Experiments on semi-supervised learning tasks demonstrate that the proposed Elastic GNNs obtain better adaptivity on benchmark datasets and are significantly robust to graph adversarial attacks. The implementation of Elastic GNNs is available at https://github.com/lxiaorui/ElasticGNN.

1. Introduction

Graph neural networks (GNNs) generalize traditional deep neural networks (DNNs) from regular grids, such as image, video, and text, to irregular data such as social networks, transportation networks, and biological networks, which are typically denoted as graphs (Defferrard et al., 2016; Kipf & Welling, 2016). One popular such generalization is the neural message passing framework (Gilmer et al., 2017):

$$\mathbf{x}_u^{(k+1)} = \text{UPDATE}^{(k)}\big(\mathbf{x}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\big) \tag{1}$$

where $\mathbf{x}_u^{(k)} \in \mathbb{R}^d$ denotes the feature vector of node u in k-th iteration of message passing and $\mathbf{m}_{\mathcal{N}(u)}^{(k)}$ is the message aggregated from u's neighborhood $\mathcal{N}(u)$. The specific architecture design has been motivated from spectral domain (Kipf & Welling, 2016; Defferrard et al., 2016) and spatial domain (Hamilton et al., 2017; Veličković et al., 2017; Scarselli et al., 2008; Gilmer et al., 2017). Recent study (Ma et al., 2020) has proven that the message passing schemes in numerous popular GNNs, such as GCN, GAT, PPNP, and APPNP, intrinsically perform the ℓ_2-based graph smoothing to the graph signal, and they can be considered as solving the graph signal denoising problem:

$$\arg\min_{\mathbf{F}} \mathcal{L}(\mathbf{F}) := \|\mathbf{F} - \mathbf{X}_{\text{in}}\|_F^2 + \lambda \,\text{tr}(\mathbf{F}^\top \mathbf{L}\mathbf{F}), \tag{2}$$

where $\mathbf{X}_{\text{in}} \in \mathbb{R}^{n\times d}$ is the input signal and $\mathbf{L} \in \mathbb{R}^{n\times n}$ is the graph Laplacian matrix encoding the graph structure. The first term guides $\mathbf{F}$ to be close to input signal $\mathbf{X}_{\text{in}}$, while the second term enforces global smoothness to the filtered signal $\mathbf{F}$. The resulted message passing schemes can be derived by different optimization solvers, and they typically entail the aggregation of node features from neighboring nodes, which intuitively coincides with the cluster or consistency assumption that neighboring nodes should be similar (Zhu & Ghahramani; Zhou et al., 2004). While existing GNNs are prominently driven by ℓ_2-based graph smoothing, ℓ_2-based methods enforce smoothness globally and the level of smoothness is usually shared across the whole graph. However, the level of smoothness over different regions of the graph can be different. For instance, node features or labels can change significantly between clusters but smoothly within the cluster (Zhu, 2005). Therefore, it is desired to enhance the local smoothness adaptivity of GNNs.

Motivated by the idea of trend filtering (Kim et al., 2009; Tibshirani et al., 2014; Wang et al., 2016), we aim to achieve the goal via ℓ_1-based graph smoothing. Intuitively, compared with ℓ_2-based methods, ℓ_1-based methods penalize large values less and thus preserve discontinuity or non-smooth signal better. Theoretically, ℓ_1-based methods tend to promote signal sparsity to trade for discontinuity (Rudin et al., 1992; Tibshirani et al., 2005; Sharpnack et al., 2012). Owning to these advantages, trend filtering (Tibshirani et al., 2014) and graph trend filter (Wang et al., 2016; Varma et al., 2019) demonstrate that ℓ_1-based graph smoothing can adapt to inhomogenous level of smoothness of signals and yield estimators with k-th order piecewise polynomial functions, such as piecewise constant, linear and quadratic functions, depending on the order of the graph difference operator.

[*]Equal contribution [1]Department of Computer Science and Engineering, Michigan State University, USA [2]School of Mathematics, Shandong University, China [3]Department of Computational Mathematics, Science and Engineering, Michigan State University, USA. Correspondence to: Xiaorui Liu <xiaorui@msu.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

While ℓ_1-based methods exhibit various appealing properties and have been extensively studied in different domains such as signal processing (Elad, 2010), statistics and machine learning (Hastie et al., 2015), it has rarely been investigated in the design of GNNs. In this work, we attempt to bridge this gap and enhance the local smoothnesss adaptivity of GNNs via ℓ_1-based graph smoothing.

Incorporating ℓ_1-based graph smoothing in the design of GNNs faces tremendous challenges. First, since the message passing schemes in GNNs can be derived from the optimization iteration of the graph signal denoising problem, a fast, efficient and scalable optimization solver is desired. Unfortunately, to solve the associated optimization problem involving ℓ_1 norm is challenging since the objective function is composed by smooth and non-smooth components and the decision variable is further coupled by the discrete graph difference operator. Second, to integrate the derived messaging passing scheme into GNNs, it has to be composed by simple operations that are friendly to the back-propagation training of the whole GNNs. Third, it requires an appropriate normalization step to deal with diverse node degrees, which is often overlooked by existing graph total variation and graph trend filtering methods. Our attempt to address these challenges leads to a family of novel GNNs, i.e., Elastic GNNs. Our key contributions can be summarized as follows:

- We introduce ℓ_1-based graph smoothing in the design of GNNs to further enhance the local smoothness adaptivity, for the first time;
- We derive a novel and general message passing scheme, i.e., Elastic Message Passing (EMP), and develop a family of GNN architectures, i.e., Elastic GNNs, by integrating the proposed message passing scheme into deep neural nets;
- Extensive experiments demonstrate that Elastic GNNs obtain better adaptivity on various real-world datasets, and they are significantly robust to graph adversarial attacks. The study on different variants of Elastic GNNs suggest that ℓ_1 and ℓ_2-based graph smoothing are complementary and the proposed GNNs are more versatile.

2. Preliminary

We use bold upper-case letters such as $\mathbf{X}$ to denote matrices and bold lower-case letters such as $\mathbf{x}$ to define vectors. Given a matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$, we use $\mathbf{X}_i$ to denote its i-th row and $\mathbf{X}_{ij}$ to denote its element in i-th row and j-th column. We define the Frobenius norm, ℓ_1 norm, and ℓ_{21} norm of matrix $\mathbf{X}$ as $\|\mathbf{X}\|_F = \sqrt{\sum_{ij} \mathbf{X}_{ij}^2}$, $\|\mathbf{X}\|_1 = \sum_{ij} |\mathbf{X}_{ij}|$, and $\|\mathbf{X}\|_{21} = \sum_i \|\mathbf{X}_i\|_2 = \sum_i \sqrt{\sum_j \mathbf{X}_{ij}^2}$, respectively. We define $\|\mathbf{X}\|_2 = \sigma_{\max}(\mathbf{X})$ where $\sigma_{\max}(\mathbf{X})$ is the largest singular value of $\mathbf{X}$. Given two matrices $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{n\times d}$, we define the inner product as $\langle \mathbf{X}, \mathbf{Y}\rangle = \mathrm{tr}(\mathbf{X}^\top \mathbf{Y})$.

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be a graph with the node set $\mathcal{V} = \{v_1, \ldots, v_n\}$ and the undirected edge set $\mathcal{E} = \{e_1, \ldots, e_m\}$. We use $\mathcal{N}(v_i)$ to denote the neighboring nodes of node v_i, including v_i itself. Suppose that each node is associated with a d-dimensional feature vector, and the features for all nodes are denoted as $\mathbf{X}_{\text{fea}} \in \mathbb{R}^{n\times d}$. The graph structure $\mathcal{G}$ can be represented as an adjacent matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, where $\mathbf{A}_{ij} = 1$ when there exists an edge between nodes v_i and v_j. The graph Laplacian matrix is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the diagonal degree matrix. Let $\Delta \in \{-1, 0, 1\}^{m\times n}$ be the oriented incident matrix, which contains one row for each edge. If $e_\ell = (i, j)$, then Δ has ℓ-th row as:

$$\Delta_\ell = (0, \ldots, \underbrace{-1}_{i}, \ldots, \underbrace{1}_{j}, \ldots, 0)$$

where the edge orientation can be arbitrary. Note that the incident matrix and unnormalized Laplacian matrix have the equivalence $\mathbf{L} = \Delta^\top \Delta$. Next, we briefly introduce some necessary background about the graph signal denoising perspective of GNNs and the graph trend filtering methods.

2.1. GNNs as Graph Signal Denoising

It is evident from recent work (Ma et al., 2020) that many popular GNNs can be uniformly understood as graph signal denoising with Laplacian smoothing regularization. Here we briefly describe several representative examples.

GCN. The message passing scheme in Graph Convolutional Networks (GCN) (Kipf & Welling, 2016),

$$\mathbf{X}_{\text{out}} = \tilde{\mathbf{A}}\mathbf{X}_{\text{in}},$$

is equivalent to one gradient descent step to minimize $\mathrm{tr}(\mathbf{F}^\top(\mathbf{I} - \tilde{\mathbf{A}})\mathbf{F})$ with the initial $\mathbf{F} = \mathbf{X}_{\text{in}}$ and stepsize $1/2$. Here $\tilde{\mathbf{A}} = \hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}$ with $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ being the adjacent matrix with self-loop, whose degree matrix is $\hat{\mathbf{D}}$.

PPNP & APPNP. The message passing scheme in PPNP and APPNP (Klicpera et al., 2018) follow the aggregation rules

$$\mathbf{X}_{\text{out}} = \alpha\big(\mathbf{I} - (1-\alpha)\tilde{\mathbf{A}}\big)^{-1}\mathbf{X}_{\text{in}},$$

and

$$\mathbf{X}^{(k+1)} = (1-\alpha)\tilde{\mathbf{A}}\mathbf{X}^{(k)} + \alpha\mathbf{X}_{\text{in}}.$$

They are shown to be the exact solution and one gradient descent step with stepsize $\alpha/2$ for the following problem

$$\min_{\mathbf{F}} \|\mathbf{F} - \mathbf{X}_{\text{in}}\|_F^2 + (1/\alpha - 1)\,\mathrm{tr}(\mathbf{F}^\top(\mathbf{I} - \tilde{\mathbf{A}})\mathbf{F}). \quad (3)$$

For more comprehensive illustration, please refer to (Ma et al., 2020). We point out that all these message passing schemes adopt ℓ_2-based graph smoothing as the signal differences between neighboring nodes are penalized by the

square of ℓ_2 norm, e.g., $\sum_{(v_i,v_j)\in\mathcal{E}} \|\frac{\mathbf{F}_i}{\sqrt{d_i+1}} - \frac{\mathbf{F}_j}{\sqrt{d_j+1}}\|_2^2$ with d_i being the node degree of node v_i. The resulted message passing schemes are usually linear smoothers which smooth the input signal by their linear transformation.

2.2. Graph Trend Filtering

In the univariate case, the k-th order graph trend filtering (GTF) estimator (Wang et al., 2016) is given by

$$\arg\min_{\mathbf{f}\in\mathbb{R}^n} = \frac{1}{2}\|\mathbf{f}-\mathbf{x}\|_2^2 + \lambda\|\Delta^{(k+1)}\mathbf{f}\|_1 \tag{4}$$

where $\mathbf{x}\in\mathbb{R}^n$ is the 1-dimensional input signal of n nodes and $\Delta^{(k+1)}$ is a k-th order graph difference operator. When $k=0$, it penalizes the absolute difference across neighboring nodes in graph $\mathcal{G}$:

$$\|\Delta^{(1)}\mathbf{f}\|_1 = \sum_{(v_i,v_j)\in\mathcal{E}} |\mathbf{f}_i - \mathbf{f}_j|$$

where $\Delta^{(1)}$ is equivalent to the incident matrix Δ. Generally, k-th order graph difference operators can be defined recursively:

$$\Delta^{(k+1)} = \begin{cases} \Delta^\top\Delta^{(k)} = \mathbf{L}^{\frac{k+1}{2}} \in \mathbb{R}^{n\times n} & \text{for odd k} \\ \Delta\Delta^{(k)} = \Delta\mathbf{L}^{\frac{k}{2}} \in \mathbb{R}^{m\times n} & \text{for even k.} \end{cases}$$

It is demonstrated that GTF can adapt to inhomogeneity in the level of smoothness of signal and tends to provide piecewise polynomials over graphs (Wang et al., 2016). For instance, when $k=0$, the sparsity induced by the ℓ_1-based penalty $\|\Delta^{(1)}\mathbf{f}\|_1$ implies that many of the differences $\mathbf{f}_i-\mathbf{f}_j$ are zeros across edges $(v_i,v_j)\in\mathcal{E}$ in $\mathcal{G}$. The piecewise property originates from the discontinuity of signal allowed by less aggressive ℓ_1 penalty, with adaptively chosen knot nodes or knot edges. Note that the smoothers induced by GTF are not linear smoothers and cannot be simply represented by linear transformation of the input signal.

3. Elastic Graph Neural Networks

In this section, we first propose a new graph signal denoising estimator. Then we develop an efficient optimization algorithm for solving the denoising problem and introduce a novel, general and efficient message passing scheme, i.e., Elastic Message Passing (EMP), for graph signal smoothing. Finally, the integration of the proposed message passing scheme and deep neural networks leads to Elastic GNNs.

3.1. Elastic Graph Signal Estimator

To combine the advantages of ℓ_1 and ℓ_2-based graph smoothing, we propose the following elastic graph signal estimator:

$$\arg\min_{\mathbf{F}\in\mathbb{R}^{n\times d}} \underbrace{\lambda_1\|\Delta\mathbf{F}\|_1}_{g_1(\Delta\mathbf{F})} + \underbrace{\frac{\lambda_2}{2}\mathrm{tr}(\mathbf{F}^\top\mathbf{L}\mathbf{F}) + \frac{1}{2}\|\mathbf{F}-\mathbf{X}_{\text{in}}\|_F^2}_{f(\mathbf{F})} \tag{5}$$

where $\mathbf{X}_{\text{in}}\in\mathbb{R}^{n\times d}$ is the d-dimensional input signal of n nodes. The first term can be written in an edge-centric way: $\|\Delta^{(1)}\mathbf{F}\|_1 = \sum_{(v_i,v_j)\in\mathcal{E}}\|\mathbf{F}_i-\mathbf{F}_j\|_1$, which penalizes the absolute difference across connected nodes in graph $\mathcal{G}$. Similarly, the second term penalizes the difference quadratically via $\mathrm{tr}(\mathbf{F}^\top\mathbf{L}\mathbf{F}) = \sum_{(v_i,v_j)\in\mathcal{E}}\|\mathbf{F}_i-\mathbf{F}_j\|_2^2$. The last term is the fidelity term which preserves the similarity with the input signal. The regularization coefficients λ_1 and λ_2 control the balance between ℓ_1 and ℓ_2-based graph smoothing.

Remark 1. *It is potential to consider higher-order graph differences in both the ℓ_1-based and ℓ_2-based smoothers. But, in this work, we focus on the 0-th order graph difference operator Δ, since we assume the piecewise constant prior for graph representation learning.*

Normalization. In existing GNNs, it is beneficial to normalize the Laplacian matrix for better numerical stability, and the normalization trick is also crucial for achieving superior performance. Therefore, for the ℓ_2-based graph smoothing, we follow the common normalization trick in GNNs: $\tilde{\mathbf{L}} = \mathbf{I} - \tilde{\mathbf{A}}$, where $\tilde{\mathbf{A}} = \hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}$, $\hat{\mathbf{A}} = \mathbf{A}+\mathbf{I}$ and $\hat{\mathbf{D}}_{ii} = d_i = \sum_j \hat{\mathbf{A}}_{ij}$. It leads to a degree normalized penalty

$$\mathrm{tr}(\mathbf{F}^\top\tilde{\mathbf{L}}\mathbf{F}) = \sum_{(v_i,v_j)\in\mathcal{E}} \left\|\frac{\mathbf{F}_i}{\sqrt{d_i+1}} - \frac{\mathbf{F}_j}{\sqrt{d_j+1}}\right\|_2^2.$$

In the literature of graph total variation and graph trend filtering, the normalization step is often overlooked and the graph difference operator is directly used as in GTF (Wang et al., 2016; Varma et al., 2019). To achieve better numerical stability and handle diverse node degrees in real-world graphs, we propose to normalize each column of the incident matrix by the square root of node degrees for the ℓ_1-based graph smoothing as follows[1]:

$$\tilde{\Delta} = \Delta\hat{\mathbf{D}}^{-\frac{1}{2}}.$$

It leads to a degree normalized total variation penalty [2]

$$\|\tilde{\Delta}\mathbf{F}\|_1 = \sum_{(v_i,v_j)\in\mathcal{E}} \left\|\frac{\mathbf{F}_i}{\sqrt{d_i+1}} - \frac{\mathbf{F}_j}{\sqrt{d_j+1}}\right\|_1.$$

Note that this normalized incident matrix maintains the relation with the normalized Laplacian matrix as in the unnormalized case

$$\tilde{\mathbf{L}} = \tilde{\Delta}^\top\tilde{\Delta} \tag{6}$$

given that

$$\tilde{\mathbf{L}} = \hat{\mathbf{D}}^{-\frac{1}{2}}(\hat{\mathbf{D}}-\hat{\mathbf{A}})\hat{\mathbf{D}}^{-\frac{1}{2}} = \hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{L}\hat{\mathbf{D}}^{-\frac{1}{2}} = \hat{\mathbf{D}}^{-\frac{1}{2}}\Delta^\top\Delta\hat{\mathbf{D}}^{-\frac{1}{2}}.$$

[1] It naturally supports read-value edge weights if the edge weights are set in the incident matrix Δ.

[2] With the normalization, the piecewise constant prior is up to the degree scaling, i.e., sparsity in $\tilde{\Delta}\mathbf{F}$.

With the normalization, the estimator defined in (5) becomes:

$$\arg\min_{\mathbf{F}\in\mathbb{R}^{n\times d}} \underbrace{\lambda_1\|\tilde{\Delta}\mathbf{F}\|_1}_{g_1(\tilde{\Delta}\mathbf{F})} + \underbrace{\frac{\lambda_2}{2}\mathrm{tr}(\mathbf{F}^\top\tilde{\mathbf{L}}\mathbf{F}) + \frac{1}{2}\|\mathbf{F}-\mathbf{X}_{\text{in}}\|_F^2}_{f(\mathbf{F})}. \tag{7}$$

Capture correlation among dimensions. The node features in real-world graphs are usually multi-dimensional. Although the estimator defined in (7) is able to handle multi-dimensional data since the signal from different dimensions are separable under ℓ_1 and ℓ_2 norm, such estimator treats each feature dimension independently and does not exploit the potential relation between feature dimensions. However, the sparsity patterns of node difference across edges could be shared among feature dimensions. To better exploit this potential correlation, we propose to couple the multi-dimensional features by ℓ_{21} norm, which penalizes the summation of ℓ_2 norm of the node difference

$$\|\tilde{\Delta}\mathbf{F}\|_{21} = \sum_{(v_i,v_j)\in\mathcal{E}} \left\| \frac{\mathbf{F}_i}{\sqrt{d_i+1}} - \frac{\mathbf{F}_j}{\sqrt{d_j+1}} \right\|_2.$$

This penalty promotes the row sparsity of $\tilde{\Delta}\mathbf{F}$ and enforces similar sparsity patterns among feature dimensions. In other words, if two nodes are similar, all their feature dimensions should be similar. Therefore, we define the ℓ_{21}-based estimator as

$$\arg\min_{\mathbf{F}\in\mathbb{R}^{n\times d}} \underbrace{\lambda_1\|\tilde{\Delta}\mathbf{F}\|_{21}}_{g_{21}(\tilde{\Delta}\mathbf{F})} + \underbrace{\frac{\lambda_2}{2}\mathrm{tr}(\mathbf{F}^\top\tilde{\mathbf{L}}\mathbf{F}) + \frac{1}{2}\|\mathbf{F}-\mathbf{X}_{\text{in}}\|_F^2}_{f(\mathbf{F})} \tag{8}$$

where $g_{21}(\cdot) = \lambda_1\|\cdot\|_{21}$. In the following subsections, we will use $g(\cdot)$ to represent both $g_1(\cdot)$ and $g_{21}(\cdot)$. We use ℓ_1 to represent both ℓ_1 and ℓ_{21} if not specified.

3.2. Elastic Message Passing

For the ℓ_2-based graph smoother, message passing schemes can be derived from the gradient descent iterations of the graph signal denoising problem, as in the case of GCN and APPNP (Ma et al., 2020). However, computing the estimators defined by (7) and (8) is much more challenging because of the nonsmoothness, and the two components, i.e., $f(\mathbf{F})$ and $g(\tilde{\Delta}\mathbf{F})$, are non-separable as they are coupled by the graph difference operator $\tilde{\Delta}$. In the literature, researchers have developed optimization algorithms for the graph trend filtering problem (4) such as Alternating Direction Method of Multipliers (ADMM) and Newton type algorithms (Wang et al., 2016; Varma et al., 2019). However, these algorithms require to solve the minimization of a non-trivial sub-problem in each single iteration, which incurs high computation complexity. Moreover, it is unclear how to make these iterations compatible with the back-propagation training of deep learning models. This motivates us to design an algorithm which is not only efficient but also friendly to back-propagation training. To this end, we propose to solve an equivalent saddle point problem using a primal-dual algorithm with efficient computations.

Saddle point reformulation. For a general convex function $g(\cdot)$, its conjugate function is defined as

$$g^*(\mathbf{Z}) := \sup_{\mathbf{X}} \langle \mathbf{Z}, \mathbf{X}\rangle - g(\mathbf{X}).$$

By using $g(\tilde{\Delta}\mathbf{F}) = \sup_{\mathbf{Z}} \langle \tilde{\Delta}\mathbf{F}, \mathbf{Z}\rangle - g^*(\mathbf{Z})$, the problem (7) and (8) can be equivalently written as the following saddle point problem:

$$\min_{\mathbf{F}} \max_{\mathbf{Z}} f(\mathbf{F}) + \langle \tilde{\Delta}\mathbf{F}, \mathbf{Z}\rangle - g^*(\mathbf{Z}). \tag{9}$$

where $\mathbf{Z} \in \mathbb{R}^{m\times d}$. Motivated by Proximal Alternating Predictor-Corrector (PAPC) (Loris & Verhoeven, 2011; Chen et al., 2013), we propose an efficient algorithm with per iteration low computation complexity and convergence guarantee:

$$\begin{cases} \bar{\mathbf{F}}^{k+1} &= \mathbf{F}^k - \gamma\nabla f(\mathbf{F}^k) - \gamma\tilde{\Delta}^\top\mathbf{Z}^k, & (10)\\ \mathbf{Z}^{k+1} &= \mathbf{prox}_{\beta g^*}(\mathbf{Z}^k + \beta\tilde{\Delta}\bar{\mathbf{F}}^{k+1}), & (11)\\ \mathbf{F}^{k+1} &= \mathbf{F}^k - \gamma\nabla f(\mathbf{F}^k) - \gamma\tilde{\Delta}^\top\mathbf{Z}^{k+1}, & (12) \end{cases}$$

where $\mathbf{prox}_{\beta g^*}(\mathbf{X}) = \arg\min_{\mathbf{Y}} \frac{1}{2}\|\mathbf{Y}-\mathbf{X}\|_F^2 + \beta g^*(\mathbf{Y})$. The stepsizes, γ and β, will be specified later. The first step (10) obtains a prediction of $\mathbf{F}^{k+1}$, i.e., $\bar{\mathbf{F}}^{k+1}$, by a gradient descent step on primal variable $\mathbf{F}^k$. The second step (11) is a proximal dual ascent step on the dual variable $\mathbf{Z}^k$ based on the predicted $\bar{\mathbf{F}}^{k+1}$. Finally, another gradient descent step on the primal variable based on $(\mathbf{F}^k, \mathbf{Z}^{k+1})$ gives next iteration $\mathbf{F}^{k+1}$ (12). Algorithm (10)–(12) can be interpreted as a "predict-correct" algorithm for the saddle point problem (9). Next we demonstrate how to compute the proximal operator in Eq. (11).

Proximal operators. Using the Moreau's decomposition principle (Bauschke & Combettes, 2011)

$$\mathbf{X} = \mathbf{prox}_{\beta g^*}(\mathbf{X}) + \lambda\mathbf{prox}_{\beta^{-1}g}(\mathbf{X}/\beta),$$

we can rewrite the step (11) using the proximal operator of $g(\cdot)$, that is,

$$\mathbf{prox}_{\beta g^*}(\mathbf{X}) = \mathbf{X} - \beta\mathbf{prox}_{\frac{1}{\beta}g}(\frac{1}{\beta}\mathbf{X}). \tag{13}$$

We discuss the two options for the function $g(\cdot)$ corresponding to the objectives (7) and (8).

$$
\begin{cases}
\mathbf{Y}^{k+1} = \gamma \mathbf{X}_{\text{in}} + (1-\gamma)\tilde{\mathbf{A}}\mathbf{F}^k \\
\bar{\mathbf{F}}^{k+1} = \mathbf{Y}^k - \gamma \tilde{\Delta}^\top \mathbf{Z}^k \\
\bar{\mathbf{Z}}^{k+1} = \mathbf{Z}^k + \beta \tilde{\Delta} \bar{\mathbf{F}}^{k+1} \\
\begin{cases}
\mathbf{Z}^{k+1} = \min(|\bar{\mathbf{Z}}^{k+1}|, \lambda_1) \cdot \operatorname{sign}(\bar{\mathbf{Z}}^{k+1}) & \textbf{(Option I: } \ell_1 \text{ norm)} \\
\mathbf{Z}_i^{k+1} = \min(\|\bar{\mathbf{Z}}_i^{k+1}\|_2, \lambda_1) \cdot \frac{\bar{\mathbf{Z}}_i^{k+1}}{\|\bar{\mathbf{Z}}_i^{k+1}\|_2}, \forall i \in [m] & \textbf{(Option II: } \ell_{21} \text{ norm)}
\end{cases} \\
\mathbf{F}^{k+1} = \mathbf{Y}^k - \gamma \tilde{\Delta}^\top \mathbf{Z}^{k+1}
\end{cases}
$$

Figure 1. Elastic Message Passing (EMP). $\mathbf{F}^0 = \mathbf{X}_{\text{in}}$ and $\mathbf{Z}^0 = \mathbf{0}^{m\times d}$.

- **Option I** (ℓ_1 norm): $g_1(\mathbf{X}) = \lambda_1 \|\mathbf{X}\|_1$

  By definition, the proximal operator of $\frac{1}{\beta} g_1(\mathbf{X})$ is

  $$\mathbf{prox}_{\frac{1}{\beta} g_1}(\mathbf{X}) = \arg\min_{\mathbf{Y}} \frac{1}{2}\|\mathbf{Y}-\mathbf{X}\|_F^2 + \frac{1}{\beta}\lambda_1 \|\mathbf{Y}\|_1,$$

  which is equivalent to the soft-thresholding operator (component-wise):

  $$\begin{aligned}(S_{\frac{1}{\beta}\lambda_1}(\mathbf{X}))_{ij} =& \operatorname{sign}(\mathbf{X}_{ij}) \max(|\mathbf{X}_{ij}| - \frac{1}{\beta}\lambda_1, 0) \\ =& \mathbf{X}_{ij} - \operatorname{sign}(\mathbf{X}_{ij}) \min(|\mathbf{X}_{ij}|, \frac{1}{\beta}\lambda_1).\end{aligned}$$

  Therefore, using (13), we have

  $$(\mathbf{prox}_{\beta g_1^*}(\mathbf{X}))_{ij} = \operatorname{sign}(\mathbf{X}_{ij}) \min(|\mathbf{X}_{ij}|, \lambda_1). \quad (14)$$

  which is a *component-wise projection* onto the ℓ_∞ ball of radius λ_1.

- **Option II** (ℓ_{21} norm): $g_{21}(\mathbf{X}) = \lambda_1 \|\mathbf{X}\|_{2,1}$

  By definition, the proximal operator of $\frac{1}{\beta} g_{21}(\mathbf{X})$ is

  $$\mathbf{prox}_{\frac{1}{\beta} g_{21}}(\mathbf{X}) = \arg\min_{\mathbf{Y}} \frac{1}{2}\|\mathbf{Y}-\mathbf{X}\|_F^2 + \frac{1}{\beta}\lambda_1 \|\mathbf{Y}\|_{2,1}$$

  with the i-th row being

  $$\left(\mathbf{prox}_{\frac{1}{\beta} g_{21}}(\mathbf{X})\right)_i = \frac{\mathbf{X}_i}{\|\mathbf{X}_i\|_2} \max(\|\mathbf{X}_i\|_2 - \frac{1}{\beta}\lambda_1, 0).$$

  Similarly, using (13), we have the i-th row of $\mathbf{prox}_{\beta g_{21}^*}(\mathbf{X})$ being

  $$\begin{aligned}&(\mathbf{prox}_{\beta g_{21}^*}(\mathbf{X}))_i \\ &= \mathbf{X}_i - \beta \mathbf{prox}_{\frac{1}{\beta} g_{21}}(\mathbf{X}_i/\beta) \\ &= \mathbf{X}_i - \beta \frac{\mathbf{X}_i/\beta}{\|\mathbf{X}_i/\beta\|_2} \max(\|\mathbf{X}_i/\beta\|_2 - \lambda_1/\beta, 0) \\ &= \mathbf{X}_i - \frac{\mathbf{X}_i}{\|\mathbf{X}_i\|_2} \max(\|\mathbf{X}_i\|_2 - \lambda_1, 0) \\ &= \frac{\mathbf{X}_i}{\|\mathbf{X}_i\|_2}(\|\mathbf{X}_i\|_2 - \max(\|\mathbf{X}_i\|_2 - \lambda_1, 0)) \\ &= \frac{\mathbf{X}_i}{\|\mathbf{X}_i\|_2} \min(\|\mathbf{X}_i\|_2, \lambda_1), \quad (15)\end{aligned}$$

  which is a *row-wise projection* on the ℓ_2 ball of radius λ_1. Note that the proximal operator in the ℓ_1 norm case treats each feature dimension independently, while in the ℓ_{21} norm case, it couples the multi-dimensional features, which is consistent with the motivation to exploit the correlation among feature dimensions.

The Algorithm (10)–(12) and the proximal operators (14) and (15) enable us to derive the final message passing scheme. Note that the computation $\mathbf{F}^k - \gamma \nabla f(\mathbf{F}^k)$ in steps (10) and (12) can be shared to save computation. Therefore, we decompose the step (10) into two steps:

$$\begin{aligned}\mathbf{Y}^k &= \mathbf{F}^k - \gamma \nabla f(\mathbf{F}^k) \\ &= \big((1-\gamma)\mathbf{I} - \gamma\lambda_2 \tilde{\mathbf{L}}\big)\mathbf{F}^k + \gamma \mathbf{X}_{\text{in}}, \quad (16) \\ \bar{\mathbf{F}}^{k+1} &= \mathbf{Y}^k - \gamma \tilde{\Delta}^\top \mathbf{Z}^k. \quad (17)\end{aligned}$$

In this work, we choose $\gamma = \frac{1}{1+\lambda_2}$ and $\beta = \frac{1}{2\gamma}$. Therefore, with $\tilde{\mathbf{L}} = \mathbf{I} - \tilde{\mathbf{A}}$, Eq. (16) can be simplified as

$$\mathbf{Y}^{k+1} = \gamma \mathbf{X}_{\text{in}} + (1-\gamma)\tilde{\mathbf{A}}\mathbf{F}^k. \quad (18)$$

Let $\bar{\mathbf{Z}}^{k+1} := \mathbf{Z}^k + \beta \tilde{\Delta} \bar{\mathbf{F}}^{k+1}$, then steps (11) and (12) become

$$\begin{aligned}\mathbf{Z}^{k+1} &= \mathbf{prox}_{\beta g^*}(\bar{\mathbf{Z}}^{k+1}), \quad (19) \\ \mathbf{F}^{k+1} &= \mathbf{F}^k - \gamma \nabla f(\mathbf{F}^k) - \gamma \tilde{\Delta} \mathbf{Z}^{k+1} \\ &= \mathbf{Y}^k - \gamma \tilde{\Delta}^\top \mathbf{Z}^{k+1}. \quad (20)\end{aligned}$$

Substituting the proximal operators in (19) with (14) and (15), we obtain the complete elastic message passing scheme (EMP) as summarized in Figure 1.

Interpretation of EMP. EMP can be interpreted as the standard message passing (MP) ($\mathbf{Y}$ in Fig. 1) with extra operations (the following steps). The extra operations compute $\tilde{\Delta}^\top \mathbf{Z}$ to adjust the standard MP such that sparsity in $\tilde{\Delta}\mathbf{F}$ is promoted and some large node differences can be preserved. EMP is general and covers some existing propagation rules as special cases as demonstrated in Remark 2.

Remark 2 (Special cases). *If there is only ℓ_2-based regularization, i.e., $\lambda_1 = 0$, then according to the projection operator, we have $\mathbf{Z}^k = \mathbf{0}^{m\times n}$. Therefore, with $\gamma = \frac{1}{1+\lambda_2}$, the proposed message passing scheme reduces to*

$$\mathbf{F}^{k+1} = \frac{1}{1+\lambda_2}\mathbf{X}_{in} + \frac{\lambda_2}{1+\lambda_2}\tilde{\mathbf{A}}\mathbf{F}^k.$$

If $\lambda_2 = \frac{1}{\alpha} - 1$, it recovers the message passing in APPNP:

$$\mathbf{F}^{k+1} = \alpha\mathbf{X}_{in} + (1-\alpha)\tilde{\mathbf{A}}\mathbf{F}^k.$$

If $\lambda_2 = \infty$, it recovers the simple aggregation operation in many GNNs:

$$\mathbf{F}^{k+1} = \tilde{\mathbf{A}}\mathbf{F}^k.$$

Computation Complexity. EMP is efficient and composed by simple operations. The major computation cost comes from four sparse matrix multiplications, include $\tilde{\mathbf{A}}\mathbf{F}^k, \tilde{\Delta}^\top\mathbf{Z}^k, \tilde{\Delta}\bar{\mathbf{F}}^{k+1}$ and $\tilde{\Delta}^\top\mathbf{Z}^{k+1}$. The computation complexity is in the order $O(md)$ where m is the number of edges in graph $\mathcal{G}$ and d is the feature dimension of input signal $\mathbf{X}_{\text{in}}$. Other operations are simple matrix additions and projection.

The convergence of EMP and the parameter settings are justified by Theorem 1, with a proof deferred to Appendix B.

Theorem 1 (Convergence). *Under the stepsize setting $\gamma < \frac{2}{1+\lambda_2\|\tilde{\mathbf{L}}\|_2}$ and $\beta \leq \frac{4}{3\gamma\|\tilde{\Delta}\tilde{\Delta}^\top\|_2}$, the elastic message passing scheme (EMP) in Figure 1 converges to the optimal solution of the elastic graph signal estimator defined in (7) (Option I) or (8) (Option II). It is sufficient to choose any $\gamma < \frac{2}{1+2\lambda_2}$ and $\beta \leq \frac{2}{3\gamma}$ since $\|\tilde{\mathbf{L}}\|_2 = \|\tilde{\Delta}^\top\tilde{\Delta}\|_2 = \|\tilde{\Delta}\tilde{\Delta}^\top\|_2 \leq 2$.*

3.3. Elastic GNNs

Incorporating the elastic message passing scheme from the elastic graph signal estimator (7) and (8) into deep neural networks, we introduce a family of GNNs, namely Elastic GNNs. In this work, we follow the decoupled way as proposed in APPNP (Klicpera et al., 2018), where we first make predictions from node features and aggregate the prediction through the proposed EMP:

$$\mathbf{Y}_{\text{pre}} = \textbf{EMP}\left(h_\theta(\mathbf{X}_{\text{fea}}), K, \lambda_1, \lambda_2\right). \tag{21}$$

$\mathbf{X}_{\text{fea}} \in \mathbb{R}^{n\times d}$ denotes the node features, $h_\theta(\cdot)$ is any machine learning model, such as multilayer perceptrons (MLPs), θ is the learnable parameters in the model, and K is the number of message passing steps. The training objective is the cross entropy loss defined by the final prediction $\mathbf{Y}_{\text{pre}}$ and labels for training data. Elastic GNNs also have the following nice properties:

- In addition to the backbone neural network model, Elastic GNNs only require to set up three hyperparameters including two coefficients λ_1, λ_2 and the propagation step K, but they do not introduce any learnable parameters. Therefore, it reduces the risk of overfitting.
- The hyperparameters λ_1 and λ_2 provide better smoothness adaptivity to Elastic GNNs depending on the smoothness properties of the graph data.
- The message passing scheme only entails simple and efficient operations, which makes it friendly to the efficient and end-to-end back-propagation training of the whole GNN model.

4. Experiment

In this section, we conduct experiments to validate the effectiveness of the proposed Elastic GNNs. We first introduce the experimental settings. Then we assess the performance of Elastic GNNs and investigate the benefits of introducing ℓ_1-based graph smoothing into GNNs with semi-supervised learning tasks under normal and adversarial settings. In the ablation study, we validate the local adaptive smoothness, sparsity pattern, and convergence of EMP.

4.1. Experimental Settings

Datasets. We conduct experiments on 8 real-world datasets including three citation graphs, i.e., Cora, Citeseer, Pubmed (Sen et al., 2008), two co-authorship graphs, i.e., Coauthor CS and Coauthor Physics (Shchur et al., 2018), two co-purchase graphs, i.e., Amazon Computers and Amazon Photo (Shchur et al., 2018), and one blog graph, i.e., Polblogs (Adamic & Glance, 2005). In Polblogs graph, node features are not available so we set the feature matrix to be a $n \times n$ identity matrix.

Baselines. We compare the proposed Elastic GNNs with representative GNNs including GCN (Kipf & Welling, 2016), GAT (Veličković et al., 2017), ChebNet (Defferrard et al., 2016), GraphSAGE (Hamilton et al., 2017), APPNP (Klicpera et al., 2018) and SGC (Wu et al., 2019). For all models, we use 2 layer neural networks with 64 hidden units.

Parameter settings. For each experiment, we report the average performance and the standard variance of 10 runs. For all methods, hyperparameters are tuned from the following search space: 1) learning rate: $\{0.05, 0.01, 0.005\}$; 2) weight decay: {5e-4, 5e-5, 5e-6}; 3) dropout rate: $\{0.5,\ 0.8\}$. For APPNP, the propagation step K is tuned from $\{5, 10\}$ and the parameter α is tuned from $\{0, 0.1, 0.2, 0.3, 0.5, 0.8, 1.0\}$. For Elastic GNNs, the propagation step K is tuned from $\{5, 10\}$ and parameters λ_1 and λ_2 are tuned from $\{0, 3, 6, 9\}$. As suggested by Theorem 1, we set $\gamma = \frac{1}{1+\lambda_2}$ and $\beta = \frac{1}{2\gamma}$ in the proposed elastic message passing scheme. Adam optimizer (Kingma & Ba, 2014) is used in all experiments.

4.2. Performance on Benchmark Datasets

On commonly used datasets including Cora, CiteSeer, PubMed, Coauthor CS, Coauthor Physics, Amazon Computers and Amazon Photo, we compare the performance of the proposed Elastic GNN ($\ell_{21} + \ell_2$) with representative GNN baselines on the semi-supervised learning task. The detail statistics of these datasets and data splits are summarized in Table 5 in Appendix A. The classification accuracy are showed in Table 1. From these results, we can make the following observations:

- Elastic GNN outperforms GCN, GAT, ChebNet, GraphSAGE and SGC by significant margins on all datasets. For instance, Elastic GNN improves over GCN by 3.1%, 2.0% and 1.8% on Cora, CiteSeer and PubMed datasets. The improvement comes from the global and local smoothness adaptivity of Elastic GNN.
- Elastic GNN ($\ell_{21} + \ell_2$) consistently achieves higher performance than APPNP on all datasets. Essentially, Elastic GNN covers APPNP as a special case when there is only ℓ_2 regularization, i.e., $\lambda_1 = 0$. Beyond the ℓ_2-based graph smoothing, the ℓ_{21}-based graph smoothing further enhances the local smoothness adaptivity. This comparison verifies the benefits of introducing ℓ_{21}-based graph smoothing in GNNs.

4.3. Robustness Under Adversarial Attack

Locally adaptive smoothness makes Elastic GNNs more robust to adversarial attack on graph structure. This is because the attack tends to connect nodes with different labels, which fuzzes the cluster structure in the graph. But EMP can tolerate large node differences along these wrong edges, and maintain the smoothness along correct edges.

To validate this, we evaluate the performance of Elastic GNNs under untargeted adversarial graph attack, which tries to degrade GNN models' overall performance by deliberately modifying the graph structure. We use the MetaAttack (Zügner & Günnemann, 2019) implemented in DeepRobust (Li et al., 2020)[3], a PyTorch library for adversarial attacks and defenses, to generate the adversarially attacked graphs based on four datasets including Cora, CiteSeer, Polblogs and PubMed. We randomly split $10\%/10\%/80\%$ of nodes for training, validation and test. The detailed data statistics are summarized in Table 6 in Appendix A. Note that following the works (Zügner et al., 2018; Zügner & Günnemann, 2019; Entezari et al., 2020; Jin et al., 2020), we only consider the largest connected component (LCC) in the adversarial graphs. Therefore, the results in Table 2 are not directly comparable with the results in Table 1. We focus on investigating the robustness introduced by ℓ_1-based graph smoothing but not on adversarial defense so we don't compare with defense strategies. Existing defense strategies can be applied on Elastic GNNs to further improve the robustness against attacks.

[3] https://github.com/DSE-MSU/DeepRobust

Variants of Elastic GNNs. To make a deeper investigation of Elastic GNNs, we consider the following variants: (1) ℓ_2 ($\lambda_1 = 0$); (2) ℓ_1 ($\lambda_2 = 0$, Option I); (3) ℓ_{21} ($\lambda_2 = 0$, Option II); (4) $\ell_1 + \ell_2$ (Option I); (5) $\ell_{21} + \ell_2$ (Option II). To save computation, we fix the learning rate as 0.01, weight decay as 0.0005, dropout rate as 0.5 and $K = 10$ since this setting works well for the chosen datasets and models. Only λ_1 and λ_2 are tuned. The classification accuracy under different perturbation rates ranging from 0% to 20% is summarized in Table 2. From the results, we can make the following observations:

- All variants of Elastic GNNs outperforms GCN and GAT by significant margins under all perturbation rates. For instance, when the pertubation rate is 15%, Elastic GNN ($\ell_{21} + \ell_2$) improves over GCN by 12.1%, 7.4%, 13.7% and 7.7% on the four datasets being considered. This is because Elastic GNN can adapt to the change of smoothness while GCN and GAT can not adapt well when the perturbation rate increases.
- ℓ_{21} outperforms ℓ_1 in most cases, and $\ell_{21} + \ell_2$ outperforms $\ell_1 + \ell_2$ in almost all cases. It demonstrates the benefits of exploiting the correlation between feature channels by coupling multi-dimensional features via ℓ_{21} norm.
- ℓ_{21} outperforms ℓ_2 in most cases, which suggests the benefits of local smoothness adaptivity. When ℓ_{21} and ℓ_2 is combined, the Elastic GNN ($\ell_{21} + \ell_2$) achieves significantly better performance than solely ℓ_2, ℓ_{21} or ℓ_1 variant in almost all cases. It suggests that ℓ_1 and ℓ_2-based graph smoothing are complementary to each other, and combining them provides significant better robustness against adversarial graph attacks.

4.4. Ablation Study

We provide ablation study to further investigate the adaptive smoothness, sparsity pattern, and convergence of EMP in Elastic GNN, based on three datasets including Cora, CiteSeer and PubMed. In this section, we fix $\lambda_1 = 3, \lambda_2 = 3$ for Elastic GNN, and $\alpha = 0.1$ for APPNP. We fix learning rate as 0.01, weight decay as 0.0005 and dropout rate as 0.5 since this setting works well for both methods.

Adaptive smoothness. It is expected that ℓ_1-based smoothing enhances local smoothness adaptivity by increasing the smoothness along correct edges (connecting nodes with same labels) while lowering smoothness along wrong edges (connecting nodes with different labels). To validate this,

Table 1. Classification accuracy (%) on benchmark datasets with 10 times random data splits.

Model	Cora	CiteSeer	PubMed	CS	Physics	Computers	Photo
ChebNet	76.3 ± 1.5	67.4 ± 1.5	75.0 ± 2.0	91.8 ± 0.4	OOM	**81.0 ± 2.0**	90.4 ± 1.0
GCN	79.6 ± 1.1	68.9 ± 1.2	77.6 ± 2.3	91.6 ± 0.6	93.3 ± 0.8	79.8 ± 1.6	90.3 ± 1.2
GAT	80.1 ± 1.2	68.9 ± 1.8	77.6 ± 2.2	91.1 ± 0.5	93.3 ± 0.7	79.3 ± 2.4	89.6 ± 1.6
SGC	80.2 ± 1.5	68.9 ± 1.3	75.5 ± 2.9	90.1 ± 1.3	93.1 ± 0.6	73.0 ± 2.0	83.5 ± 2.9
APPNP	82.2 ± 1.3	70.4 ± 1.2	78.9 ± 2.2	**92.5 ± 0.3**	93.7 ± 0.7	80.1 ± 2.1	90.8 ± 1.3
GraphSAGE	79.0 ± 1.1	67.5 ± 2.0	77.6 ± 2.0	91.7 ± 0.5	92.5 ± 0.8	80.7 ± 1.7	90.9 ± 1.0
ElasticGNN	**82.7 ± 1.0**	**70.9 ± 1.4**	**79.4 ± 1.8**	**92.5 ± 0.3**	**94.2 ± 0.5**	80.7 ± 1.8	**91.3 ± 1.3**

Table 2. Classification accuracy (%) under different perturbation rates of adversarial graph attack.

Dataset	Ptb Rate	Basic GNN		Elastic GNN				
		GCN	GAT	ℓ_2	ℓ_1	ℓ_{21}	$\ell_1 + \ell_2$	$\ell_{21} + \ell_2$
Cora	0%	83.5±0.4	84.0±0.7	**85.8±0.4**	85.1±0.5	85.3±0.4	**85.8±0.4**	**85.8±0.4**
	5%	76.6±0.8	80.4±0.7	81.0±1.0	**82.3±1.1**	81.6±1.1	81.9±1.4	82.2±0.9
	10%	70.4±1.3	75.6±0.6	76.3±1.5	76.2±1.4	77.9±0.9	78.2±1.6	**78.8±1.7**
	15%	65.1±0.7	69.8±1.3	72.2±0.9	73.3±1.3	75.7±1.2	76.9±0.9	**77.2±1.6**
	20%	60.0±2.7	59.9±0.6	67.7±0.7	63.7±0.9	70.3±1.1	67.2±5.3	**70.5±1.3**
Citeseer	0%	72.0±0.6	73.3±0.8	73.6±0.9	73.2±0.6	73.2±0.5	73.6±0.6	**73.8±0.6**
	5%	70.9±0.6	72.9±0.8	72.8±0.5	72.8±0.5	72.8±0.5	**73.3±0.6**	72.9±0.5
	10%	67.6±0.9	70.6±0.5	70.2±0.6	70.8±0.6	70.7±1.2	72.4±0.9	**72.6±0.4**
	15%	64.5±1.1	69.0±1.1	70.2±0.6	68.1±1.4	68.2±1.1	71.3±1.5	**71.9±0.7**
	20%	62.0±3.5	61.0±1.5	**64.9±1.0**	64.7±0.8	64.7±0.8	64.7±0.8	64.7±0.8
Polblogs	0%	95.7±0.4	95.4±0.2	95.4±0.2	**95.8±0.3**	**95.8±0.3**	**95.8±0.3**	**95.8±0.3**
	5%	73.1±0.8	83.7±1.5	82.8±0.3	78.7±0.6	78.7±0.7	82.8±0.4	**83.0±0.3**
	10%	70.7±1.1	76.3±0.9	73.7±0.3	75.2±0.4	75.3±0.7	81.5±0.2	**81.6±0.3**
	15%	65.0±1.9	68.8±1.1	68.9±0.9	72.1±0.9	71.5±1.1	77.8±0.9	**78.7±0.5**
	20%	51.3±1.2	51.5±1.6	65.5±0.7	68.1±0.6	68.7±0.7	77.4±0.2	**77.5±0.2**
Pubmed	0%	87.2±0.1	83.7±0.4	**88.1±0.1**	86.7±0.1	87.3±0.1	**88.1±0.1**	**88.1±0.1**
	5%	83.1±0.1	78.0±0.4	**87.1±0.2**	86.2±0.1	87.0±0.1	**87.1±0.2**	**87.1±0.2**
	10%	81.2±0.1	74.9±0.4	86.6±0.1	86.0±0.2	86.9±0.2	86.3±0.1	**87.0±0.1**
	15%	78.7±0.1	71.1±0.5	85.7±0.2	85.4±0.2	86.4±0.2	85.5±0.1	**86.4±0.2**
	20%	77.4±0.2	68.2±1.0	85.8±0.1	85.4±0.1	86.4±0.1	85.4±0.1	**86.4±0.1**

we compute the average adjacent node differences (based on node features in the last layer) along wrong and correct edges separately, and use the ratio between these two averages to measure the smoothness adaptivity. The results are summarized in Table 3. It is clearly observed that for all datasets, the ratio for ElasticGNN is significantly higher than ℓ_2 based method such as APPNP, which validates its better local smoothness adaptivity.

Sparsity pattern. To validate the piecewise constant property enforced by EMP, we also investigate the sparsity pattern in the adjacent node differences, i.e., $\tilde{\Delta}\mathbf{F}$, based on node features in the last layer. Node difference along edge e_i is defined as sparse if $\|(\tilde{\Delta}\mathbf{F})_i\|_2 < 0.1$. The sparsity ratios for ℓ_2-based method such as APPNP and ℓ_1-based method such as Elastic GNN are summarized in Table 4. It can be observed that in Elastic GNN, a significant portion of $\tilde{\Delta}\mathbf{F}$ are sparse for all datasets. While in APPNP, this portion is much smaller. This sparsity pattern validates the piecewise constant prior as designed.

Table 3. Ratio between average node differences along wrong and correct edges.

Model	Cora	CiteSeer	PubMed
ℓ_2 (APPNP)	1.57	1.35	1.43
ℓ_{21}+ℓ_2 (ElasticGNN)	2.03	1.94	1.79

Table 4. Sparsity ratio (i.e., $\|(\tilde{\Delta}\mathbf{F})_i\|_2 < 0.1$) in node differences $\tilde{\Delta}\mathbf{F}$.

Model	Cora	CiteSeer	PubMed
ℓ_2 (APPNP)	2%	16%	11%
ℓ_{21}+ℓ_2 (ElasticGNN)	37%	74%	42%

Convergence of EMP. We provide two additional experiments to demonstrate the impact of propagation step K on classification performance and the convergence of message passing scheme. Figure 2 shows that the increase of classifi-

cation accuracy when the propagation step K increases. It verifies the effectiveness of EMP in improving graph representation learning. It also shows that a small number of propagation step can achieve very good performance, and therefore the computation cost for EMP can be small. Figure 3 shows the decreasing of the objective value defined in Eq. (8) during the forward message passing process, and it verifies the convergence of the proposed EMP as suggested by Theorem 1.

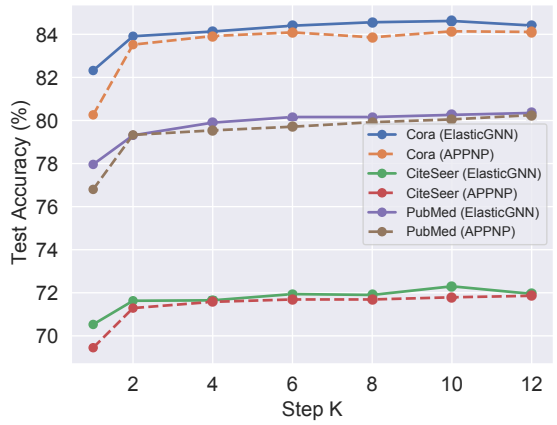

Figure 2. Classification accuracy under different propagation steps.

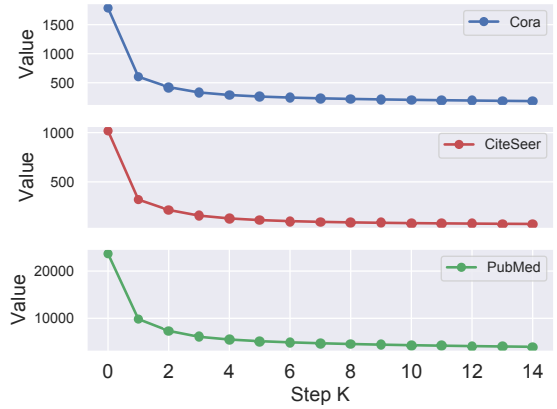

Figure 3. Convergence of the objective value for the problem in Eq. (8) during message passing.

5. Related Work

The design of GNN architectures can be majorly motivated in spectral domain (Kipf & Welling, 2016; Defferrard et al., 2016) and spatial domain (Hamilton et al., 2017; Veličković et al., 2017; Scarselli et al., 2008; Gilmer et al., 2017). The message passing scheme (Gilmer et al., 2017; Ma & Tang, 2020) for feature aggregation is one central component of GNNs. Recent works have proven that the message passing in GNNs can be regarded as low-pass graph filters (Nt & Maehara, 2019; Zhao & Akoglu, 2019). Generally, it is recently proved that message passing in many GNNs can be unified in the graph signal denosing framework (Ma et al., 2020; Pan et al., 2020; Zhu et al., 2021; Chen et al., 2020). We point out that they intrinsically perform ℓ_2-based graph smoothing and typically can be represented as linear smoothers.

ℓ_1-based graph signal denoising has been explored in graph trend filtering (Wang et al., 2016; Varma et al., 2019) which tends to provide estimators with k-th order piecewise polynomials over graphs. Graph total variation has also been utilized in semi-supervised learning (Nie et al., 2011; Jung et al., 2016; Jung et al., 2019; Aviles-Rivero et al., 2019), spectral clustering (Bühler & Hein, 2009; Bresson et al., 2013b) and graph cut problems (Szlam & Bresson, 2010; Bresson et al., 2013a). However, it is unclear whether these algorithms can be used to design GNNs. To the best of our knowledge, we make first such investigation in this work.

6. Conclusion

In this work, we propose to enhance the smoothness adaptivity of GNNs via ℓ_1 and ℓ_2-based graph smoothing. Through the proposed elastic graph signal estimator, we derive a novel, efficient and general message passing scheme, i.e., elastic message passing (EMP). Integrating the proposed message passing scheme and deep neural networks leads to a family of GNNs, i.e., Elastic GNNs. Extensitve experiments on benchmark datasets and adversarially attacked graphs demonstrate the benefits of introducing ℓ_1-based graph smoothing in the design of GNNs. The empirical study suggests that ℓ_1 and ℓ_2-based graph smoothing is complementary to each other, and the proposed Elastic GNNs has better smoothnesss adaptivity owning to the integration of ℓ_1 and ℓ_2-based graph smoothing. We hope the proposed elastic message passing scheme can inspire more powerful GNN architecture design in the future.

Acknowledgements

This research is supported by the National Science Foundation (NSF) under grant numbers CNS-1815636, IIS-1928278, IIS-1714741, IIS-1845081, IIS-1907704, IIS-1955285, and Army Research Office (ARO) under grant number W911NF-21-1-0198. Ming Yan is supported by NSF grant DMS-2012439 and Facebook Faculty Research Award (Systems for ML).

References

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

Aviles-Rivero, A. I., Papadakis, N., Li, R., Alsaleh, S. M.,

Tan, R. T., and Schonlieb, C.-B. When labelled data hurts: Deep semi-supervised classification with the graph 1-laplacian. *arXiv preprint arXiv:1906.08635*, 2019.

Bauschke, H. H. and Combettes, P. L. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer Publishing Company, Incorporated, 1st edition, 2011. ISBN 1441994661.

Bresson, X., Laurent, T., Uminsky, D., and von Brecht, J. H. An adaptive total variation algorithm for computing the balanced cut of a graph, 2013a.

Bresson, X., Laurent, T., Uminsky, D., and Von Brecht, J. H. Multiclass total variation clustering. *arXiv preprint arXiv:1306.1185*, 2013b.

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.

Chen, P., Huang, J., and Zhang, X. A primal–dual fixed point algorithm for convex separable minimization with applications to image restoration. *Inverse Problems*, 29 (2):025011, 2013.

Chen, S., Eldar, Y. C., and Zhao, L. Graph unrolling networks: Interpretable neural networks for graph signal denoising, 2020.

Chung, F. R. and Graham, F. C. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 3844–3852, 2016.

Elad, M. *Sparse and redundant representations: from theory to applications in signal and image processing*. Springer Science & Business Media, 2010.

Entezari, N., Al-Sayouri, S. A., Darvishzadeh, A., and Papalexakis, E. E. All you need is low (rank) defending against adversarial attacks on graphs. In *WSDM*, 2020.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pp. 1263–1272. PMLR, 2017.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*, 2017.

Hastie, T., Tibshirani, R., and Wainwright, M. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman Hall/CRC, 2015. ISBN 1498712169.

Jin, W., Ma, Y., Liu, X., Tang, X., Wang, S., and Tang, J. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 66–74, 2020.

Jung, A., Hero III, A. O., Mara, A., and Jahromi, S. Semi-supervised learning via sparse label propagation. *arXiv preprint arXiv:1612.01414*, 2016.

Jung, A., Hero, III, A. O., Mara, A. C., Jahromi, S., Heimowitz, A., and Eldar, Y. C. Semi-supervised learning in network-structured data via total variation minimization. *IEEE Transactions on Signal Processing*, 67(24): 6256–6269, 2019. doi: 10.1109/TSP.2019.2953593.

Kim, S.-J., Koh, K., Boyd, S., and Gorinevsky, D. ℓ_1 trend filtering. *SIAM review*, 51(2):339–360, 2009.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Klicpera, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

Li, Y., Jin, W., Xu, H., and Tang, J. Deeprobust: A pytorch library for adversarial attacks and defenses, 2020.

Li, Z. and Yan, M. A primal-dual algorithm with optimal stepsizes and its application in decentralized consensus optimization. *arXiv preprint arXiv:1711.06785*, 2017.

Loris, I. and Verhoeven, C. On a generalization of the iterative soft-thresholding algorithm for the case of non-separable penalty. *Inverse Problems*, 27(12):125007, 2011.

Ma, Y. and Tang, J. *Deep Learning on Graphs*. Cambridge University Press, 2020.

Ma, Y., Liu, X., Zhao, T., Liu, Y., Tang, J., and Shah, N. A unified view on graph neural networks as graph signal denoising. *arXiv preprint arXiv:2010.01777*, 2020.

Nie, F., Wang, H., Huang, H., and Ding, C. Unsupervised and semi-supervised learning via ℓ_1-norm graph. In *2011 International Conference on Computer Vision*, pp. 2268–2273. IEEE, 2011.

Nt, H. and Maehara, T. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

Pan, X., Shiji, S., and Gao, H. A unified framework for convolution-based graph neural networks. *https://openreview.net/forum?id=zUMD–Fb9Bt*, 2020.

Rudin, L. I., Osher, S., and Fatemi, E. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

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.

Sharpnack, J., Singh, A., and Rinaldo, A. Sparsistency of the edge lasso over graphs. In *Artificial Intelligence and Statistics*, pp. 1028–1036. PMLR, 2012.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Szlam, A. and Bresson, X. Total variation, cheeger cuts. In *ICML*, 2010.

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., and Knight, K. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

Tibshirani, R. J. et al. Adaptive piecewise polynomial estimation via trend filtering. *Annals of statistics*, 42(1): 285–323, 2014.

Varma, R., Lee, H., Kovačević, J., and Chi, Y. Vector-valued graph trend filtering with non-convex penalties. *IEEE Transactions on Signal and Information Processing over Networks*, 6:48–62, 2019.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Wang, Y.-X., Sharpnack, J., Smola, A. J., and Tibshirani, R. J. Trend filtering on graphs. *Journal of Machine Learning Research*, 17:1–41, 2016.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. PMLR, 2019.

Zhao, L. and Akoglu, L. Pairnorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations*, 2019.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. Learning with local and global consistency. 2004.

Zhu, M., Wang, X., Shi, C., Ji, H., and Cui, P. Interpreting and unifying graph neural networks with an optimization framework, 2021.

Zhu, X. and Ghahramani, Z. Learning from labeled and unlabeled data with label propagation.

Zhu, X. J. Semi-supervised learning literature survey. 2005.

Zügner, D. and Günnemann, S. Adversarial attacks on graph neural networks via meta learning. *arXiv preprint arXiv:1902.08412*, 2019.

Zügner, D., Akbarnejad, A., and Günnemann, S. Adversarial attacks on neural networks for graph data. In *KDD*. ACM, 2018.

Appendix for Elastic Graph Neural Networks

A. Data Statistics

The data statistics for the benchmark datasets used in Section 4.2 are summarized in Table 5. The data statistics for the adversarially attacked graph used in Section 4.3 are summarized in Table 6.

Table 5. Statistics of benchmark datasets.

Dataset	Classes	Nodes	Edges	Features	Training Nodes	Validation Nodes	Test Nodes
Cora	7	2708	5278	1433	20 per class	500	1000
CiteSeer	6	3327	4552	3703	20 per class	500	1000
PubMed	3	19717	44324	500	20 per class	500	1000
Coauthor CS	15	18333	81894	6805	20 per class	30 per class	Rest nodes
Coauthor Physics	5	34493	247962	8415	20 per class	30 per class	Rest nodes
Amazon Computers	10	13381	245778	767	20 per class	30 per class	Rest nodes
Amazon Photo	8	7487	119043	745	20 per class	30 per class	Rest nodes

Table 6. Dataset Statistics for adversarially attacked graph.

	$\mathbf{N_{LCC}}$	$\mathbf{E_{LCC}}$	Classes	Features
Cora	2,485	5,069	7	1,433
CiteSeer	2,110	3,668	6	3,703
Polblogs	1,222	16,714	2	/
PubMed	19,717	44,338	3	500

B. Convergence Guarantee

We provide Theorem 1 to show the convergence guarantee of the proposed elastic messsage passing scheme and the practical guidance for parameter settings in EMP.

Theorem 1 (Convergence of EMP)**.** *Under the stepsize setting* $\gamma < \frac{2}{1+\lambda_2\|\tilde{\mathbf{L}}\|_2}$ *and* $\beta \leq \frac{4}{3\gamma\|\tilde{\Delta}\tilde{\Delta}^\top\|_2}$*, the elastic message passing scheme (EMP) in Figure 1 converges to the optimal solution of the elastic graph signal estimator defined in* (7) *(Option I) or* (8) *(Option II). It is sufficient to choose any* $\gamma < \frac{2}{1+2\lambda_2}$ *and* $\beta \leq \frac{2}{3\gamma}$ *since* $\|\tilde{\mathbf{L}}\|_2 = \|\tilde{\Delta}^\top\tilde{\Delta}\|_2 = \|\tilde{\Delta}\tilde{\Delta}^\top\|_2 \leq 2$.

Proof. We first consider the general problem

$$\min_{\mathbf{F}} f(\mathbf{F}) + g(\mathbf{B}\mathbf{F}) \tag{22}$$

where f and g are convex functions and $\mathbf{B}$ is a bounded linear operator. It is proved in (Loris & Verhoeven, 2011; Chen et al., 2013) that the iterations in (10)–(12) guarantee the convergence of $\mathbf{F}^k$ to the optimal solution of the minimization problem (22) if the parameters satisfy $\gamma < \frac{2}{L}$ and $\beta \leq \frac{1}{\gamma\lambda_{\max}(\mathbf{B}\mathbf{B}^\top)}$, where L is the Lipschitz constant of $\nabla f(\mathbf{F})$. These conditions are further relaxed to $\gamma < \frac{2}{L}$ and $\beta \leq \frac{4}{3\gamma\lambda_{\max}(\mathbf{B}\mathbf{B}^\top)}$ in (Li & Yan, 2017).

For the specific problems defined in (7) and (8), the two function components f and g are both convex, and the linear operator Δ is bounded. The Lipschitz constant of $\nabla f(\mathbf{F})$ can be computed by the largest eigenvalue of the Hessian matrix of $f(\mathbf{F})$:

$$L = \lambda_{\max}(\nabla^2 f(\mathbf{F})) = \lambda_{\max}(\mathbf{I} + \lambda_2\tilde{\mathbf{L}}) = 1 + \lambda_2\|\tilde{\mathbf{L}}\|_2.$$

Therefore, the elastic message passing scheme derived from iterations (10)–(12) is guaranteed to converge to the optimal solution of problem (7) (Option I) or problem (8) (Option II) if the stepsizes satisfy $\gamma < \frac{2}{1+\lambda_2\|\tilde{\mathbf{L}}\|_2}$ and $\beta \leq \frac{4}{3\gamma\|\tilde{\Delta}\tilde{\Delta}^\top\|_2}$.

Let $\tilde{\Delta} = \mathbf{U}\Sigma\mathbf{V}^\top$ be the singular value decomposition of $\tilde{\Delta}$ and we derive

$$\|\tilde{\Delta}\tilde{\Delta}^\top\|_2 = \|\mathbf{U}\Sigma\mathbf{V}^\top\mathbf{V}\Sigma\mathbf{U}^\top\|_2 = \|\mathbf{U}\Sigma^2\mathbf{U}^\top\|_2 = \|\mathbf{V}\Sigma^2\mathbf{V}^\top\|_2 = \|\mathbf{V}\Sigma\mathbf{U}^\top\mathbf{U}\Sigma\mathbf{V}^\top\|_2 = \|\tilde{\Delta}^\top\tilde{\Delta}\|_2.$$

The equivalence $\tilde{\mathbf{L}} = \tilde{\Delta}^\top\tilde{\Delta}$ in (6) further gives

$$\|\tilde{\mathbf{L}}\|_2 = \|\tilde{\Delta}^\top\tilde{\Delta}\|_2 = \|\tilde{\Delta}\tilde{\Delta}^\top\|_2.$$

Since $\|\tilde{\mathbf{L}}\|_2 \leq 2$ (Chung & Graham, 1997), we have $\frac{2}{1+2\lambda_2} \leq \frac{2}{1+\lambda_2\|\tilde{\mathbf{L}}\|_2}$ and $\frac{2}{3\gamma} \leq \frac{4}{3\gamma\|\tilde{\Delta}\tilde{\Delta}^\top\|_2}$. Therefore, $\gamma < \frac{2}{1+2\lambda_2}$ $\beta \leq \frac{2}{3\gamma}$ are sufficient for the convergence of EMP.

□