STABILITY OF GRAPH NEURAL NETWORKS TO RELATIVE PERTURBATIONS

Fernando Gama, Alejandro Ribeiro[*]

University of Pennsylvania
Dept. of Electrical and Systems Engineering
Philadelphia, PA

Joan Bruna[†]

New York University
Courant Institute of Mathematical Sciences
New York, NY

ABSTRACT

Graph neural networks (GNNs), consisting of a cascade of layers applying a graph convolution followed by a pointwise nonlinearity, have become a powerful architecture to process signals supported on graphs. Graph convolutions (and thus, GNNs), rely heavily on knowledge of the graph for operation. However, in many practical cases the graph shift operator (GSO) is not known and needs to be estimated, or might change from training time to testing time. In this paper, we are set to study the effect that a change in the underlying graph topology that supports the signal has on the output of a GNN. We prove that graph convolutions with integral Lipschitz filters lead to GNNs whose output change is bounded by the size of the relative change in the topology. Furthermore, we leverage this result to show that the main reason for the success of GNNs is that they are stable architectures capable of discriminating features on high eigenvalues, which is a feat that cannot be achieved by linear graph filters (which are either stable or discriminative, but cannot be both). Finally, we comment on the use of this result to train GNNs with increased stability and run experiments on movie recommendation systems.

Index Terms— graph neural networks, graph signal processing, network data, stability, graph convolutions

1. INTRODUCTION

Networks such as power grids [1], transportation networks [2] or weather sensor networks [3] generate data with an irregular structure dictated by the topology of the network. This data can be modeled as a graph signal by assigning each entry to a node in some underlying given graph that describes the network [4]. The graph shift operator (GSO) is a linear map between graph signals where the output value at each node is a weighted average of the input values at neighboring nodes. The GSO is thus any matrix that respects the sparsity of the graph (adjacency [5], Laplacian matrix [6], etc.), and the output is said to be a *shifted* version of the input.

The operation of graph convolution, defined as a linear combination of shifted version of the signal, is used to compute the output of graph filters in an efficient and decentralized fashion [7, 8]. Furthermore, graph convolutions are used to build graph neural networks (GNNs), as a cascade of layers each of which applies a graph convolution, followed by a pointwise nonlinearity [9–11]. GNNs offer a nonlinear transformation of the input data that has achieved remarkable performance in wireless networks [12], decentralized control of robot swarms [13] and recommendation systems [14], among others [15, 16]. Graph filters and GNNs rely heavily on the knowledge of the GSO. But if we do not know the graph and need to estimate it [17], or if the graph changes with time [18], or if we want to train on one graph but test on another (transfer learning) [19], then it is of utmost importance to characterize how graph filters and GNNs react to changes in the underlying graph support.

[*]Supported by NSF CCF 1717120, ARO W911NF1710438, ARL DCIST CRA W911NF-17-2-0181, ISTC-WAS and Intel DevCloud.

[†]Partially supported by the Alfred P. Sloan Foundation, NSF RI-1816753, NSF CAREER CIF 1845360, and Samsung Electronics.

In this paper, we start by considering GSOs that are permutations of each other and prove that graph convolutions are unaffected by these node relablings (permutation equivariance). Then, we prove that for graph convolutions computed on two arbitrary GSOs, the output will differ in a manner proportional to the relative distance between the GSOs (stability to relative perturbations). These results show that there is a trade-off between stability and discriminability for linear graph filters, but that this trade-off can be overcome by the use of pointwise nonlinearities. This renders GNNs both stable and discriminative, a feat that cannot be achieved by linear graph filters.

Stability results in graph neural networks have only been developed, so far, for graph scattering transforms, which involve carefully designed, non-trainable filters [19–21]. In [20], stability to permutations is studied, as well as to perturbations on the eigenvalues and eigenvectors of the underlying graph support. In [21] graph perturbations are measured in terms of the diffusion distance, while in [19] different graph wavelets are compared in their stability [22, 23].

In Sec. 2 we introduce the GSP framework, define graph convolutions, and prove stability for linear graph filters. In Sec. 3 we prove stability for GNNs and discuss how the conditions imposed on filters determine a trade-off between discriminability and stability, which can be overcome by the inclusion of pointwise nonlinearities. Finally, in Sec. 4 we show how to train a GNN while controlling for its stability and run experiments on a movie recommendation problem. Conclusions are drawn in Sec. 5.

2. STABILITY OF GRAPH FILTERS

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ be an undirected graph with a set of N nodes $\mathcal{V}$, a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and an edge weight function $\mathcal{W} : \mathcal{E} \to \mathbb{R}_+$. This graph acts as the underlying support for the available data $\mathbf{x} \in \mathbb{R}^N$. That is, $\mathbf{x}$ is modeled as a *graph signal* where each entry $[\mathbf{x}]_n = x_n$ corresponds to the data value assigned to node n [5, 6]. The data $\mathbf{x}$ is related to the underlying graph support by means of a linear map between graph signals $\mathbf{S} : \mathbb{R}^N \to \mathbb{R}^N$ that we denote a *graph shift operator* (GSO) [4]. The GSO is a linear operator $\mathbf{S}$ that updates the data value on each node by a weighted average of the values at neighboring nodes, i.e. it *shifts* the signal across the graph. Therefore, the GSO can be written as a $N \times N$ matrix that respects the sparsity of the graph, $[\mathbf{S}]_{ij} = s_{ij} = 0$ if $i \neq j$ and $(j, i) \notin \mathcal{E}$, and the value of the output signal at node i is

$$[\mathbf{S}\mathbf{x}]_i = \sum_{j=1}^{N} [\mathbf{S}]_{ij} [\mathbf{x}]_j = \sum_{j \in \mathcal{N}_i} s_{ij} x_j \tag{1}$$

where $\mathcal{N}_i = \{j \in \mathcal{V} : (j, i) \in \mathcal{E}\}$ is the set of neighboring nodes of i, and the last equality follows from the sparsity pattern of matrix $\mathbf{S}$. Examples of GSO typically used in the literature include the adjacency matrix [5], the Laplacian matrix [6] and the Markov matrix [24].

To process data $\mathbf{x}$ in a manner that takes into account the irregular structure imposed by the underlying graph we need operations built on (1). In this light, we define the *graph convolution* as a linear combination of shifted versions of the signal [7]

$$\mathbf{y} = \sum_{k=0}^{K-1} h_k \mathbf{S}^k \mathbf{x} = \mathbf{H}(\mathbf{S})\mathbf{x} \tag{2}$$

Authorized licensed use limited to: New York University. Downloaded on June 02,2021 at 20:37:19 UTC from IEEE Xplore. Restrictions apply.

where $\mathbf{h} = [h_0, \ldots, h_{K-1}] \in \mathbb{R}^K$ is a set of K filter taps, each one weighing the information located at the k-hop neighborhood. We say that $\mathbf{H}(\mathbf{S})$ is a *graph filter* [7]. We note that (2) can be calculated by means of $K-1$ exchanges of information with the one-hop neighborhood. Also, (2) boils down to regular convolution when modeling time signals as being supported on a directed cycle, see [11] for details.

Graph filters exhibit the key property of *permutation equivariance*. Define $\mathcal{P} = \{\mathbf{P} \in \{0,1\}^{N \times N} : \mathbf{P1} = \mathbf{1}, \mathbf{P}^\mathsf{T}\mathbf{1} = \mathbf{1}\}$ the set of all permutation matrices. We prove the following.

Theorem 1 (Permutation equivariance)**.** Let $\mathbf{S}$ be the GSO of a graph $\mathcal{G}$ and $\hat{\mathbf{S}} = \mathbf{P}^\mathsf{T}\mathbf{S}\mathbf{P}$ be the GSO of a permuted version of $\mathcal{G}$. Likewise, consider signals $\mathbf{x}$ and $\hat{\mathbf{x}} = \mathbf{P}^\mathsf{T}\mathbf{x}$. Then,

$$\mathbf{H}(\hat{\mathbf{S}})\hat{\mathbf{x}} = \mathbf{P}^\mathsf{T}\mathbf{H}(\mathbf{S})\mathbf{x}. \tag{3}$$

Proof. See [25, Appendix A]. □

Theorem 1 essentially states that a graph filter on a permuted graph, applied to a correspondingly permuted signal, yields an output that is a permuted version of the original output. The immediate consequence of this theorem, is that graph filters are invariant to node relabelings.

We are ultimately interested in the effect that a more general perturbation of the GSO has on the output of a graph filter with fixed filter taps. Given two GSOs $\mathbf{S}$ and $\hat{\mathbf{S}}$, consider the distance between filters $\mathbf{H}(\mathbf{S})$ and $\mathbf{H}(\hat{\mathbf{S}})$ to be

$$\|\mathbf{H}(\mathbf{S}) - \mathbf{H}(\hat{\mathbf{S}})\|_{\mathcal{P}} = \min_{\mathbf{P}\in\mathcal{P}} \max_{\mathbf{x}:\|\mathbf{x}\|=1} \|\mathbf{P}^\mathsf{T}\mathbf{H}(\mathbf{S})\mathbf{x} - \mathbf{H}(\mathbf{P}^\mathsf{T}\hat{\mathbf{S}}\mathbf{P})\mathbf{P}^\mathsf{T}\mathbf{x}\|. \tag{4}$$

We note that (4) is the operator norm, modulo permutations. We also note that if $\hat{\mathbf{S}} = \mathbf{P}^\mathsf{T}\mathbf{S}\mathbf{P}$, then $\|\mathbf{H}(\mathbf{S}) - \mathbf{H}(\hat{\mathbf{S}})\|_{\mathcal{P}} = 0$ in virtue of Thm. 1. To measure the size of the GSO perturbation, we denote by $\mathcal{E}(\mathbf{S}, \hat{\mathbf{S}}) = \{\mathbf{E} : \mathbf{P}^\mathsf{T}\mathbf{S}\mathbf{P} = \mathbf{S} + (\mathbf{E}\mathbf{S} + \mathbf{S}\mathbf{E}), \mathbf{P} \in \mathcal{P}\}$ the set of relative errror matrix and define the distance between $\mathbf{S}$ and $\hat{\mathbf{S}}$ as

$$d(\mathbf{S}, \hat{\mathbf{S}}) = \min_{\mathbf{E}\in\mathcal{E}(\mathbf{S},\hat{\mathbf{S}})} \|\mathbf{E}\|. \tag{5}$$

In essence, given a set of filter taps $\mathbf{h}$, we want to determine how much $\|\mathbf{H}(\mathbf{S}) - \mathbf{H}(\hat{\mathbf{S}})\|_{\mathcal{P}}$ changes in relation to $d(\mathbf{S}, \hat{\mathbf{S}})$.

We can use a frequency analysis to separate the action of any given filter on a signal into the effects of the specific filter taps and that of the underlying graph support. Let $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\mathsf{H}$ be the eigendecomposition of the GSO, with $\mathbf{V}$ the eigenvector basis $\{\mathbf{v}_n\}_{n=1}^N$ and $\mathbf{\Lambda}$ a diagonal matrix containing the eigenvalues $\{\lambda_n\}_{n=1}^N$. The *graph Fourier transform* (GFT) $\tilde{\mathbf{x}}$ of a signal $\mathbf{x}$ is computed as its projection onto the eigenvector basis of the GSO, $\tilde{\mathbf{x}} = \mathbf{V}^\mathsf{H}\mathbf{x}$ [26]. Then, the GFT of the output of a graph filter $\mathbf{y} = \mathbf{H}(\mathbf{S})\mathbf{x}$ becomes

$$\tilde{\mathbf{y}} = \mathbf{V}^\mathsf{H}\big(\mathbf{H}(\mathbf{S})\mathbf{x}\big) = \sum_{k=0}^{\infty} h_k \mathbf{\Lambda}^k \tilde{\mathbf{x}} = h(\mathbf{\Lambda})\tilde{\mathbf{x}}. \tag{6}$$

where the function $h : \mathbb{R} \to \mathbb{R}$ is called the filter's *frequency response*

$$h(\lambda) = \sum_{k=0}^{\infty} h_k \lambda^k. \tag{7}$$

We note that since h is an analytic function, its application to a matrix is well defined. From (6) we see that the effect of the filter on the ith frequency coefficient is given by $\tilde{y}_i = h(\lambda_i)\tilde{x}_i$. That is, the frequency content of $\mathbf{x}$ at the ith eigenvalue, gets modified by $h(\lambda_i)$. This depends, on one hand, on the specific filter taps that determine the frequency response $h(\lambda)$ and, on the other hand, on the specific GSO under consideration that instantiates the frequency response on its eigenvalue λ_i. We thus note that the frequency response (7) is independent of the specific graph support, and only depends on the filter taps $\mathbf{h}$.

We can control the impact of changes in the GSO by carefully designing the frequency response (7). In particular, we focus on filters that are *integral Lipschitz*, see Fig. 1a.

Definition 1 (Integral Lipschitz filters)**.** Given filter taps $\mathbf{h}$ and frequency response $h(\lambda)$ [cf. (7)], we say that the corresponding filter is *integral Lipschitz* if it satisfies that $|h(\lambda)| \leq 1$ and there exists a constant $C > 0$ such that for all λ_1, λ_2 it holds that

$$\big|h(\lambda_2) - h(\lambda_1)\big| \leq C\frac{|\lambda_2 - \lambda_1|}{|\lambda_1 + \lambda_2|/2}. \tag{8}$$

Integral Lipschitz filters are those whose frequency response is Lipschitz with a constant that depends on the midpoint value of the interval. Alternatively, see that filters that satisfy (8) also satisfy $|\lambda h'(\lambda)| \leq C$, where h' is the derivative of h. These are filters that can vary arbitrarily fast for $\lambda \approx 0$, but have to be constant for $\lambda \to \infty$. We also note that this condition is reminiscent of the scale invariance of wavelet transforms [27, Ch. 7] [22, 23].

For filters that are integral Lipschitz, we can prove the following stability result.

Theorem 2 (Stability of graph filters)**.** Let $\mathbf{S}$ and $\hat{\mathbf{S}}$ be two GSOs such that $d(\mathbf{S}, \hat{\mathbf{S}}) \leq \varepsilon$ where the error matrix $\mathbf{E} \in \mathcal{E}(\mathbf{S}, \hat{\mathbf{S}})$ has an eigendecomposition $\mathbf{E} = \mathbf{U}\mathbf{M}\mathbf{U}^\mathsf{H}$. Consider a given set of filter taps $\mathbf{h}$ of an integral Lipschitz filter with constant C. Then,

$$\big\|\mathbf{H}(\mathbf{S}) - \mathbf{H}(\hat{\mathbf{S}})\big\|_{\mathcal{P}} \leq 2C\left(1 + \delta\sqrt{N}\right)\varepsilon + \mathcal{O}(\varepsilon^2) \tag{9}$$

with $\delta := (\|\mathbf{U} - \mathbf{V} + 1)^2 - 1$ measuring the eigenvector basis misalingment.

Proof. See [25, Appendix C]. □

Theorem 2 shows that the change at the output of a graph filter due to changes in the underlying GSO is proportional to the distance between those GSOs [cf. (5)]. The proportionality constant can be analyzed in two separate parts. First, we have the integral Lipschitz constant C which can be controlled by careful design of the frequency response (filter taps). Second, $(1 + \delta\sqrt{N})$ depends on the specific family of perturbations and worsens as the graph grows larger. This second part cannot be controlled and is dependent on the specific perturbations the graph support will be subject to. However, we can impose a structural constraint on the perturbation matrix $\mathbf{E}$ to obtain a constant that only depends on C. See [25, Thm. 4] for details.

3. STABILITY OF GRAPH NEURAL NETWORKS

A graph neural network (GNN) is a nonlinear map $\mathbf{\Phi}(\mathbf{S}, \mathbf{x})$ that is applied to the input $\mathbf{x}$ and takes into account the underlying graph by means of the GSO $\mathbf{S}$. It consists of a cascade of L layers, each of them applying a graph filter $\mathbf{H}_\ell(\mathbf{S})$ followed by a pointwise nonlinearity σ_ℓ (activation function)

$$\mathbf{x}_\ell = \sigma_\ell\big(\mathbf{H}_\ell(\mathbf{S})\mathbf{x}_{\ell-1}\big) \tag{10}$$

for $\ell = 1, \ldots, L$, where $\mathbf{x}_0 = \mathbf{x}$ the input signal, and $\mathbf{\Phi}(\mathbf{S}, \mathbf{x}) = \mathbf{x}_L$ the output of the last layer [9–11].

The use of graph filters (2) as $\mathbf{H}_\ell(\mathbf{S})$ means that GNNs inherit the properties of permutation equivariance and stability to relative perturbations from them.

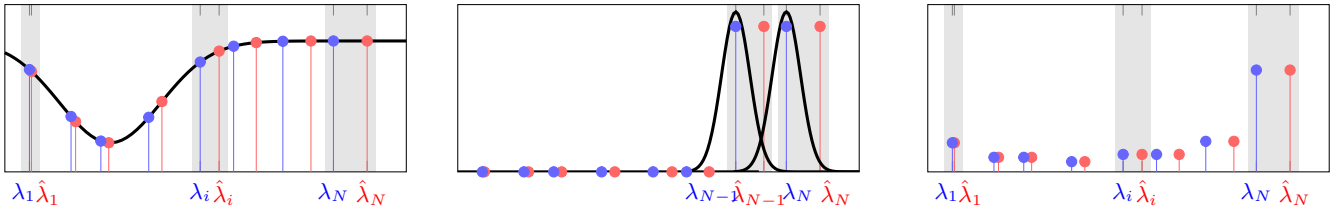
Theorem 3 (Permutation equivariance of GNNs)**.** Let $\mathbf{S}$ be the GSO of a graph $\mathcal{G}$ and $\hat{\mathbf{S}} = \mathbf{P}^\mathsf{T}\mathbf{S}\mathbf{P}$ be the GSO of a permuted version of $\mathcal{G}$. Likewise, consider signals $\mathbf{x}$ and $\hat{\mathbf{x}} = \mathbf{P}^\mathsf{T}\mathbf{x}$. Then,

$$\mathbf{\Phi}(\hat{\mathbf{S}}, \hat{\mathbf{x}}) = \mathbf{P}^\mathsf{T}\mathbf{\Phi}(\mathbf{S}, \mathbf{x}). \tag{11}$$

Proof. See [25, Appendix D]. □

Result in Thm. 3 follows through because the nonlinearities are pointwise and therefore applied separately to each node, bearing no effect on the permutation equivariance from graph filters. We note that there

Authorized licensed use limited to: New York University. Downloaded on June 02,2021 at 20:37:19 UTC from IEEE Xplore. Restrictions apply.



(a) Integral Lipschitz Filter (b) High eigenvalue features (c) Frequency mixing

Fig. 1. (a) Frequency response for an integral Lipschitz filter (in black), eigenvalues for $\mathbf{S}$ (in blue) and eigenvalues for $\hat{\mathbf{S}}$ (in red). Note that larger eigenvalues exhibit a larger change. (b) Separating energy located at λ_{N-1} from that at λ_N requires filters with sharp transitions that are not integral Lipschitz. Then, a change in eigenvalues renders these filters useless (they are not stable) (c) Applying a ReLU to a signal with all its energy located at λ_N results in a signal with energy spread through the spectrum. Information on low eigenvalues can be discriminated in a stable fashion.

are local activation functions involving neighboring exchanges that also preserve the permutation equivariance property [14].

For the stability result to hold, we need to use pointwise nonlinearities that are *normalized* Lipschitz, i.e. Lipschitz functions with constant equal to 1, $|\sigma_\ell(b) - \sigma_\ell(a)| \leq |b-a|$ for all $b, a \in \mathbb{R}$, and for all ℓ. We note that typically used activation functions like ReLU or $\tanh$ satisfy this condition.

Theorem 4 (Stability of GNNs)**.** Let $\mathbf{S}$ and $\hat{\mathbf{S}}$ be two GSOs such that $d(\mathbf{S}, \hat{\mathbf{S}}) \leq \varepsilon$ where the error matrix $\mathbf{E} \in \mathcal{E}(\mathbf{S}, \hat{\mathbf{S}})$ has an eigendecomposition $\mathbf{E} = \mathbf{U}\mathbf{M}\mathbf{U}^{\mathsf{H}}$. Consider a GNN $\mathbf{\Phi}$ with L layers where, in each layer, σ_ℓ is Lipschitz with constant 1 and the filter $\mathbf{h}_\ell$ is integral Lipschitz with constant C_ℓ. Then,

$$\left\| \mathbf{\Phi}(\mathbf{S}, \cdot) - \mathbf{\Phi}(\hat{\mathbf{S}}, \cdot) \right\|_{\mathcal{P}} \leq 2C \left(1 + \delta \sqrt{N}\right) L\varepsilon + \mathcal{O}(\varepsilon^2) \tag{12}$$

with $C = \max_\ell \{C_\ell\}$ and $\delta := (\|\mathbf{U} - \mathbf{V} + 1)^2 - 1$ measuring the eigenvector basis misalingment.

Proof. See [25, Appendix E]. □

Thm. 4 proves that GNNs are stable in the sense that, if the constitutive filters are integral Lipschitz and the activation function is normalized Lipschitz, then a change of ε in the GSOs causes a change proportional to ε in the output of the GNNs. The proportionality constant has the term C that depends on the filter design, and the term $(1 + \delta\sqrt{N})$ that depends on the specific perturbation under consideration. But it also has a constant factor L that depends on the depth of the architecture. Therefore, the deeper a GNN is the less stable it becomes. This is due to how the errors propagate and amplify through subsequent applications of graph filters.

We have proven that graph filters have the properties of permutation equivariance and stability to relative graph perturbations, and that GNNs inherit these properties. We have also observed that the corresponding graph filters have to be integral Lipschitz for stability to hold, and that the integral Lipschitz constant controls the level of stability. This observation helps explain why GNNs exhibit better performance when dealing with signals with relevant high-eigenvalue frequency content. To see this, consider the following example.

Let $\mathbf{S}$ be the GSO of a given graph, and consider the perturbation $\hat{\mathbf{S}}$ to be an edge dilation

$$\hat{\mathbf{S}} = (1 + \varepsilon)\mathbf{S} \tag{13}$$

where all edges are increased proportionally by a factor of ε. Clearly, $\mathbf{E} = (\varepsilon/2)\mathbf{I}$ and $d(\mathbf{S}, \hat{\mathbf{S}}) = \|\mathbf{E}\| \leq \varepsilon$. The eigenvalues are now $\hat{\lambda}_n = (1+\varepsilon)\lambda_n$ while the eigenvectors remain the same. We note that, even if ε is very small, the change in eigenvalues could be large if λ_n is large, see Fig. 1a.

To account for this variability in large eigenvalues (even for small ε) we need to design the frequency response [cf. (7)] so as to absorb these changes. Otherwise, the output of filtering [cf. (6)] could change significantly (the values of $h(\lambda_n)$ and $h(\hat{\lambda}_n)$ could be very different), making it unstable. The integral Lipschitz condition on filters, precisely avoids this problem by forcing the frequency response to be flat for large eigenvalues, see Fig. 1a.

The cost to pay for stability, however, is that integral Lipschitz filters are not able to discriminate information located at higher eigenvalues (Fig. 1b). In essence, linear filters are either stable or discriminative, but cannot be both. GNNs on the other hand, incorporate pointwise nonlinearities at the output of the linear graph filter. This nonlinear operation has a frequency mixing effect by which the energy of the signal is spilled throughout the spectrum, see Fig. 1c. Then, the energy that appears in smaller eigenvalues can be arbitrarily discriminated, providing GNNs a way to stably discriminate signals with large eigenvalue content. Thus, GNNs are information processing architectures that are both stable and selective.

Finally, we remark that the above proofs and analysis can be readily extended to multi-feature signals (graph signal tensors) which assign a vector of F features to each node, instead of a single scalar. Please, refer to [25] for details on the multi-feature setting.

4. NUMERICAL EXPERIMENTS

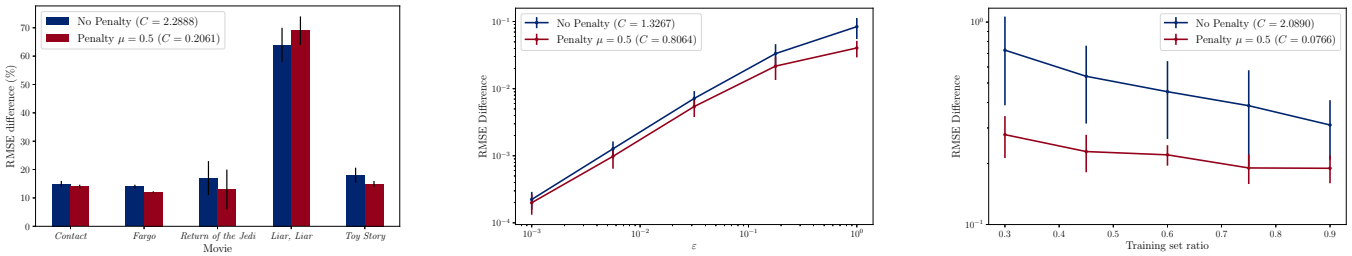
Consider a given dataset of input-output pairs $\mathcal{T} = \{(\mathbf{x}, \mathbf{y})\}$ where $\mathbf{x}$ is a graph signal defined on a graph with GSO $\mathbf{S}$. We want to use a GNN as a nonlinear map between the input $\mathbf{x}$ and the output $\mathbf{y}$. We can thus use the given dataset to *fit* or *train* the neural network by finding the filter taps $\mathbf{h}_\ell$ that minimize some loss function $\min_{\mathbf{h}_\ell} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{T}} \mathcal{L}[\mathbf{\Phi}(\mathbf{S}, \mathbf{x}), \mathbf{y}]$. We note that, in the context of training, the permutation equivariance property of GNNs serves as a form of *data augmentation*. More precisely, by exploiting the topological symmetries of the underlying graph, the GNN can learn how to process the signal on all those parts of the graph that are topologically symmetric by seeing a sample in only one of them.

By minimizing the loss function, we obtain a given set of filter taps that are a good fit for the data at hand. However, the resulting frequency responses might not have a good stability constant. To overcome this, we add a penalty to the loss function in order to control the value of the stability constant

$$\{\mathbf{h}_\ell\} = \operatorname*{argmin}_{\mathbf{h}_\ell} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{T}} \mathcal{L}\big[\mathbf{\Phi}(\mathbf{S}, \mathbf{x}), \mathbf{y}\big] + \mu \max_{\lambda \in [\lambda_a, \lambda_b]} |\lambda h'(\lambda)|. \tag{14}$$

We note that the bounds of the interval $[\lambda_a, \lambda_b]$ have to be set before training begins. One option is to set it to the eigenvalue interval of the given GSO (this would demand an eigendecomposition, albeit only once before training begins). Alternatively, we can exploit well-known bounds relating the eigenvalues with the topology of the graph [28, 29]. Furthermore, we note that the computation of the derivative $h'(\lambda)$ is straightforward, since it is also a polynomial with the same filter taps $\mathbf{h}_\ell$ that we are optimizing over. Finally, we remark that, by tweaking the penalty value μ we adjust the trade-off between better performance and more stability. If μ is too big, then the training would tend to set all filter taps to 0, which is the trivial but perfectly stable solution.

Authorized licensed use limited to: New York University. Downloaded on June 02,2021 at 20:37:19 UTC from IEEE Xplore. Restrictions apply.



(a) Train on 'Star Wars', test on other movies

(b) Synthetic perturbations

(c) Change in train/test ratio

Fig. 2. (a) RMSE degradation (in percentage) when the architectures are trained to predict the rating of the movie *Star Wars*, but are tested on 5 other movies. We see that, except for the case of *Liar, Liar*, the RMSE degradation is below 20% with the more stable architecture having a degradation below 15%. (b) RMSE degradation when testing on GSOs $\hat{\mathbf{S}}$ that have been synthetically perturbed within a relative distance of ε. The more stable architecture exhibits a smaller RMSE degradation. (c) RMSE degradation when testing the architectures on different GSOs obtained by changing the training/test ratio. When we have a ratio similar to that on which the architecture was trained, the RMSE degradation is lower.

In what follows, we consider the problem of movie recommendation systems [30]. We are given a dataset of user ratings for some movies, and we want to *learn* how a user would rate a specific movie given their previous ratings and all other users in the dataset. To do this, we build a graph where each node is a movie, and the edges are based on the Pearson correlation coefficient obtained from the pool of users that have rated any given pair of movies. See [30, Sec. II] for details on the construction of this graph. We then prune this graph keeping only the 10 nearest neighbors, we make it undirected by keeping the average edge weight and we adopt the resulting adjacency matrix as the GSO $\mathbf{S}$. Additionally, we model each user in the dataset as a graph signal $\mathbf{x}$, where the value $[\mathbf{x}]_n = x_n$ at node n is the rating that the user has given to movie n. The ratings are integers between 1 and 5, and if no rating was given, then we assign $x_n = 0$ to said node.

We consider the MovieLens-100k [31] dataset, that consists of $100,000$ ratings given by 943 users to $1,582$ movies. Following the above described model, this implies a dataset of 943 graph signals defined over a graph with $1,582$ nodes. We focus on *learning* the rating that any given user would give to a specific movie (node n) based on the ratings given to other movies (the graph signal $\mathbf{x}$) and the relationship with other users that have similar taste (given by the graph support $\mathbf{S}$). We consider all users that have rated the specific movie (have nonzero value $[\mathbf{x}]_n = x_n > 0$), take the rating x_n as the label y associated to the signal $\mathbf{x}$ and then zero-out the nth entry $[\mathbf{x}]_n = 0$ (to render the rating unknown). Specifically, we choose to estimate the rating at the movie *Star Wars* given that is the one with the largest number of ratings. We use 90% of the resulting dataset for training and 10% for testing (no samples in the test set are included when estimating the graph).

The map between the graph signal $\mathbf{x}$ (ratings for some of the movies) and the target $y = x_n$ (rating for the specific movie) is parametrized by a single-layer GNN with $F_1 = 64$ output features, using graph filters with $K_1 = 5$ filter taps, followed by a ReLU nonlinearity. Since the output of this GNN $\mathbf{x}_1$ is another graph signal, we focus particularly on the value of the 64 features at the node n of interest. We further learn a readout layer consisting of a linear combination of the resulting 64 features at node n so that the final output is a single scalar predicting the rating given at said node. We note that all operations involved are local. First, a graph convolutional layer involving the application of the graph filter bank that demands $K_1 - 1 = 4$ exchanges with the one-hop neighborhood, and then a readout layer consisting of a linear transformation of the 64 resulting features at the single node of interest (i.e. no involvement of values at any other node in this readout layer).

We train this GNN by minimizing the loss function (14), where $\mathcal{L}$ is a smooth L1 loss. We consider two different training cases leading to two different models. The first one in which there is no penalty ($\mu = 0$) and the second one where we set $\mu = 0.5$. We use the ADAM optimizer [32] with learning rate 0.005 and forgetting factors $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We train for 40 epochs using batches of size 5. In all subsequent experiments, we report averages over 10 different dataset split realizations (the split is selected at random) and the corresponding standard deviation.

For the first experiment, we train the GNNs to estimate the rating of the movie *Star Wars* both with no penalty ($\mu = 0$) and with stability penalty ($\mu = 0.5$). At test time, we obtain an RMSE of 0.8640 (± 0.1674) for the No Penalty GNN, and 0.8655 (± 0.1655) for the Penalty GNN. We then proceed to test these already trained GNNs on estimating the rating at some other movies, as shown on Fig. 2a. We see that, except for the case of the movie *Liar, liar*, the RMSE degradation for estimating the rating of a movie the GNN was not trained for is below 20%. Moreover, in all cases, the more stable GNN (the one trained with the penalty) exhibits a degradation below 15% and always better than the No Penalty GNN.

For the second experiment, we introduce a synthetic relative perturbation to the GSO $\mathbf{S}$ by randomly generating a GSO $\hat{\mathbf{S}}$ such that $d(\mathbf{S}, \hat{\mathbf{S}}) \leq \varepsilon$ [cf. (5)]. We then test on $\hat{\mathbf{S}}$ the GNNs trained on $\mathbf{S}$ for estimating the rating at node n and compare the RMSE with that obtained when testing on $\mathbf{S}$. The results illustrated in Fig. 2b show that the Penalty GNN is more stable than the No Penalty one, and that the gap in RMSE difference increases as the perturbation increases.

Finally, we consider perturbations arising from the randomness in choosing the training/test set split. Since we use the resulting training set to create $\mathbf{S}$, a change in the training set selection results in a different $\hat{\mathbf{S}}$. In Fig. 2c we show the RMSE difference between testing on the trained $0.9/0.1$ partition and testing on other partitions. Again, we observe that the Penalty GNN is more stable.

5. CONCLUSIONS

In this work we discussed the stability properties of graph filters and GNNs. We proved that both are permutation equivariant and are stable to relative perturbations of the underlying graph support. We observed that a condition for stability is that graph filters need to have a flat frequency response at large eigenvalues. We then argued that this prevents graph filters to be able to discriminate features located on these eigenvalues, and that this is a fundamental limitation of graph filters, which can thus be either stable or selective, but not both. GNNs, on the other hand, use the frequency mixing effect of nonlinearities to spread the information content throughout the eigenvalue spectrum, especially on lower eigenvalues where it can be separated in a stable fashion. Thus, GNNs are information processing architectures that are both discriminative and stable. The improved performance of GNNs over graph filters thus becomes more marked when processing signals where the relevant information content is in high frequencies. Finally, we run experiments on a movie recommendation problem, where we show that more stable architectures exhibit a better performance when tested on different conditions than the ones they were trained on.

Authorized licensed use limited to: New York University. Downloaded on June 02,2021 at 20:37:19 UTC from IEEE Xplore. Restrictions apply.

6. REFERENCES

[1] D. Owerko, F. Gama, and A. Ribeiro, "Predicting power outages using graph neural networks," in *2018 IEEE Global Conf. Signal and Inform. Process.*, Anaheim, CA, 26-29 Nov. 2018, pp. 743–747, IEEE.

[2] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *6th Int. Conf. Learning Representations*, Vancouver, BC, 30 Apr.-3 May 2018, pp. 1–16, Assoc. Comput. Linguistics.

[3] E. Isufi, A. Loukas, N. Perraudin, and G. Leus, "Forecasting time series with varma recursions on graphs," *IEEE Trans. Signal Process.*, vol. 67, no. 18, pp. 4870–4885, Sep. 2019.

[4] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, May 2018.

[5] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, Apr. 2013.

[6] D. I Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[7] S. Segarra, A. G. Marques, and A. Ribeiro, "Optimal graph-filter design and applications to distributed linear network operators," *IEEE Trans. Signal Process.*, vol. 65, no. 15, pp. 4117–4131, Aug. 2017.

[8] M. Coutino, E. Isufi, and G. Leus, "Advances in distributed graph filtering," *IEEE Trans. Signal Process.*, vol. 67, no. 9, pp. 2320–2333, May 2019.

[9] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and deep locally connected networks on graphs," in *2nd Int. Conf. Learning Representations*, Banff, AB, 14-16 Apr. 2014, pp. 1–14, Assoc. Comput. Linguistics.

[10] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *30th Conf. Neural Inform. Process. Syst.*, Barcelona, Spain, 5-10 Dec. 2016, pp. 3844–3858, Neural Inform. Process. Foundation.

[11] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, "Convolutional neural network architectures for signals supported on graphs," *IEEE Trans. Signal Process.*, vol. 67, no. 4, pp. 1034–1049, Feb. 2019.

[12] M. Eisen and A. Ribeiro, "Optimal wireless resource allocation with random edge graph neural networks," *arXiv:1909.01865v2 [eess.SP]*, 3 Oct. 2019.

[13] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro, "Learning decentralized controllers for robot swarms with graph neural networks," in *Conf. Robot Learning 2019*, Osaka, Japan, 30 Oct.-1 Nov. 2019, Int. Found. Robotics Res.

[14] L. Ruiz, F. Gama, A. G. Marques, and A. Ribeiro, "Invariance-preserving localized activation functions for graph neural networks," *arXiv:1903.12575v1 [eess.SP]*, 29 March 2019.

[15] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th Int. Conf. Learning Representations*, Toulon, France, 24-26 Apr. 2017, Assoc. Comput. Linguistics.

[16] L. Ruiz, F. Gama, and A. Ribeiro, "Gated graph convolutional recurrent neural networks," in *27th Eur. Signal Process. Conf.*, A Coruña, Spain, 2-6 Sep. 2019, Eur. Assoc. Signal Process.

[17] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology inference from spectral templates," *IEEE Trans. Signal, Inform. Process. Networks*, vol. 3, no. 3, pp. 467–483, Sep. 2017.

[18] F. Gama, E. Isufi, A. Ribeiro, and G. Leus, "Controllability of bandlimited graph processes over random time-varying graphs," *IEEE Trans. Signal Process.*, 7 Oct. 2019.

[19] F. Gama, J. Bruna, and A. Ribeiro, "Stability of Graph Scattering Transforms," in *33rd Conf. Neural Inform. Process. Syst.*, Vancouver, BC, 8-14 Dec. 2019, Neural Inform. Process. Syst. Foundation.

[20] D. Zou and G. Lerman, "Graph convolutional neural networks via scattering," *arXiv:1804.00099v2 [cs.IT]*, 18 Nov. 2018.

[21] F. Gama, A. Ribeiro, and J. Bruna, "Diffusion scattering transforms on graphs," in *7th Int. Conf. Learning Representations*, New Orleans, LA, 6-9 May 2019, pp. 1–12, Assoc. Comput. Linguistics.

[22] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmonic Anal.*, vol. 30, no. 2, pp. 129–150, March 2011.

[23] D. I. Shuman, C. Wiesmeyr, N. Holighaus, and P. Vandergheynst, "Spectrum-adapted tight graph wavelet and vertex-frequency frames," *IEEE Trans. Signal Process.*, vol. 63, no. 16, pp. 4223–4235, Aug. 2015.

[24] A. Heimowitz and Y. C. Eldar, "A unified view of diffusion maps and signal processing on graphs," in *2017 Int. Conf. Sampling Theory and Appl.*, Tallin, Estonia, 3-7 July 2017, IEEE.

[25] F. Gama, J. Bruna, and A. Ribeiro, "Stability properties of graph neural networks," *arXiv:1905.04497v2 [cs.LG]*, 4 Sep. 2019.

[26] A. Sandyhaila and J. M. F. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3042–3054, June 2014.

[27] I. Daubechies, *Ten Lectures on Wavelets*, vol. 61 of *CBMS-NSF Regional Conf. Series Appl. Math.*, SIAM, Philadelphia, PA, 1992.

[28] D. M. Cvetković, M. Doob, and H. Sachs, *Spectra of Graphs: Theory and Applications*, Academic Press, New York, NY, 1979.

[29] C. Godsil and G. Royle, *Algebraic Graph Theory*, vol. 207 of *Graduate Texts in Mathematics*, Springer, New York, NY, 2001.

[30] W. Huang, A. G. Marques, and A. Ribeiro, "Rating prediction via graph signal processing," *IEEE Trans. Signal Process.*, vol. 66, no. 19, pp. 5066–5081, Oct. 2018.

[31] F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," *ACM Trans. Interactive Intell. Syst.*, vol. 5, no. 4, pp. 19:(1–19), Jan. 2016.

[32] D. P. Kingma and J. L. Ba, "ADAM: A method for stochastic optimization," in *3rd Int. Conf. Learning Representations*, San Diego, CA, 7-9 May 2015, pp. 1–15, Assoc. Comput. Linguistics.

Authorized licensed use limited to: New York University. Downloaded on June 02,2021 at 20:37:19 UTC from IEEE Xplore. Restrictions apply.