Untrained graph neural networks for denoising

Samuel Rey*, Santiago Segarra†, Reinhard Heckel‡, and Antonio G. Marques*

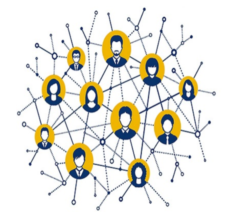
*Dept. of Signal Theory and Communications, King Juan Carlos University, Madrid, Spain

†Dept. of ECE, Rice University, Houston, USA ‡Dept. of ECE, Technical University of Munich, Munich, Germany

Why graph signal denoising?

- Data is becoming **heterogeneous** and **pervasive** [Kolaczyk09][Leskovec20]
  - ⇒ Large amounts of data are propelling the development of data-driven methods
- Growing complexity of modern systems & networks also demands new methods
  - ⇒ Popular approach: 1) Interpret the data as signals defined on a graph; and
  - ⇒ 2) Harness the graph topology to deal with irregular structure (e.g. via graph NNs)

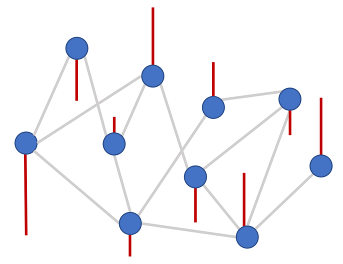

Brain network


Social network

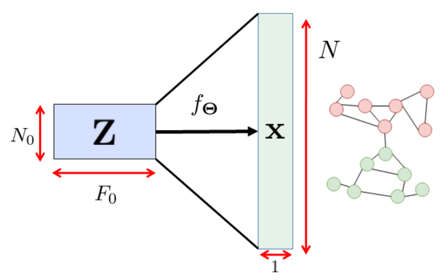

Home automation network

- **Problem:** data is corrupted with noise that may render the data useless
- **This work:** design non-linear NN architectures to remove the noise from graph signals

Graphs, graph signals, and GNNs



- Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes and adjacency $\mathbf{A} \in \mathbb{R}^{N \times N}$
  - ⇒ A_{ij} = Proximity between i and j
- Define a signal $\mathbf{x} \in \mathbb{R}^N$ on top of the graph
  - ⇒ x_i = Signal value at node i
- Graph filters are defined as $\mathbf{H} = \sum_{r=0}^{R-1} h_r \mathbf{A}^r$ [Segarra17]
- We represent a graph NN (**GNN**) as a parametric function $f_{\boldsymbol{\Theta}}(\mathbf{Z}|\mathcal{G}) : \mathbb{R}^{N^{(0)} \times F^{(0)}} \to \mathbb{R}^N$
  - ⇒ Focus on mappings from fixed input $\mathbf{Z}$ to $\mathbf{x}$



- With $\mathbf{Y}^{(0)} = \mathbf{Z}$, a GNN with L layers is given by

$$\hat{\mathbf{Y}}^{(\ell)} = \mathcal{T}^{(\ell)}_{\boldsymbol{\Theta}^{(\ell)}}\left\{\mathbf{Y}^{(\ell-1)}|\mathcal{G}\right\}, \quad 1 \le \ell \le L$$

$$Y_{ij}^{(\ell)} = g^{(\ell)}\left(\hat{Y}_{ij}^{(\ell)}\right), \quad 1 \le \ell \le L$$

  - ⇒ Graph-aware linear operator and non-linearity
- Interest on designing the **linear operator** $\mathcal{T}^{(\ell)}_{\boldsymbol{\Theta}^{(\ell)}}$ to exploit the information encoded in $\mathcal{G}$

Problem formulation and goal

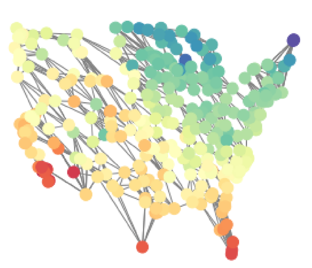
- Graph signal denoising aims at removing the noise from the observed signal
  - ⇒ Recover **unknown** signal $\mathbf{x}_0 \in \mathbb{R}^N$ from noisy observation $\mathbf{x} = \mathbf{x}_0 + \mathbf{n}$
- Traditional methods based on solving a regularized LS problem [Chen14][Wang15]

$$\hat{\mathbf{x}}_0 = \operatorname{argmin}_{\mathbf{x}_0} \|\mathbf{x} - \mathbf{x}_0\|_2^2 + \alpha R(\mathbf{x}_0|\mathcal{G})$$

  - ⇒ The graph-related regularization promotes desired properties on $\hat{\mathbf{x}}_0$
- **Our goal**: design and analyze untrained GNNs to denoise graph signals

$$\hat{\boldsymbol{\Theta}} = \operatorname{argmin}_{\boldsymbol{\Theta}} \frac{1}{2}\|\mathbf{x} - f_{\boldsymbol{\Theta}}(\mathbf{Z}|\mathcal{G})\|_2^2$$

  - ⇒ Each $\hat{\mathbf{x}}_0 = f_{\hat{\boldsymbol{\Theta}}}(\mathbf{Z}|\mathcal{G})$ is estimated individually from a single observation
  - ⇒ Weights $\hat{\boldsymbol{\Theta}}$ fitted for each $\mathbf{x}$ without training phase [Heckel20]
- **Key assumption:** GNN is designed to **learn the signal faster** than noise
  - ⇒ The GNN incorporates an implicit regularization ⇒ How to account for $\mathcal{G}$
  - ⇒ Apply SGD in combination with **early stopping**
  - ⇒ **Contribution**: Two different GNN denoising architectures (**GCG** and **GDec**)


True signal $\mathbf{x}_0$


Noisy signal $\mathbf{x}$

Graph Convolutional Generator (GCG)

- The GCG includes the graph topology via vertex-based convolutions
  - ⇒ The graph convolution operation performed via **fixed GFs** $\mathbf{H}^{(\ell)} \in \mathbb{R}^{N \times N}$
- The output of a GCG with L layers is given by the following recursion

$$\mathbf{Y}^{(\ell)} = \mathrm{ReLU}(\mathbf{H}^{(\ell)}\mathbf{Y}^{(\ell-1)}\boldsymbol{\Theta}^{(\ell)}), \quad \text{for } \ell = 1, ..., L-1$$
$$\mathbf{y}^{(L)} = \mathbf{H}^{(\ell)}\mathbf{Y}^{(L-1)}\boldsymbol{\Theta}^{(L)}$$

  - ⇒ The fixed graph filters $\mathbf{H}^{(\ell)}$ capture prior knowledge of $\mathbf{x}_0$
  - ⇒ The learnable parameters $\boldsymbol{\Theta}^{(\ell)} \in \mathbb{R}^{F^{(\ell-1)} \times F^{(\ell)}}$ mix the columns

Features of the architecture

- The GCG layer is a generalization of the GCNN layer [Kipf16]

$$\hat{\mathbf{Y}}^{(\ell)} = (\mathbf{A} + \mathbf{I})\mathbf{Y}^{(\ell-1)}\boldsymbol{\Theta}^{(\ell)} = \tilde{\mathbf{H}}\mathbf{Y}^{(\ell-1)}\boldsymbol{\Theta}^{(\ell)}$$

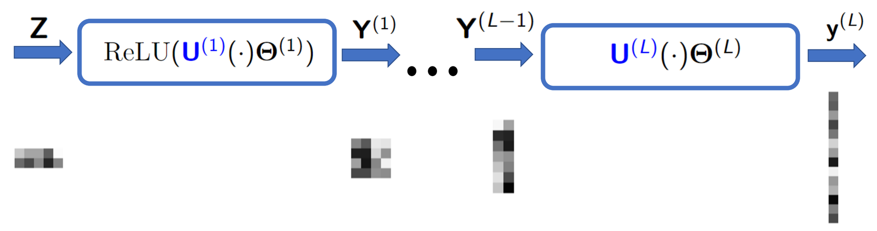
- The GCG addresses important limitations of the previous GCNN
  - ⇒ The depth of GCG and the radius of $\mathbf{H}$ are independent
  - ⇒ Avoids over-smoothing problem

Graph Decoder (GDec)

- The GDec includes the graph topology via graph upsampling
  - ⇒ We need to design the graph upsampling operators $\mathbf{U}^{(\ell)} \in \mathbb{R}^{N^{(\ell)} \times N^{(\ell-1)}}$
- The output of a GDec with L layers is given by the following recursion

$$\mathbf{Y}^{(\ell)} = \mathrm{ReLU}(\mathbf{U}^{(\ell)}\mathbf{Y}^{(\ell-1)}\boldsymbol{\Theta}^{(\ell)}), \quad \text{for } \ell = 1, ..., L-1$$
$$\mathbf{y}^{(L)} = \mathbf{U}^{(L)}\mathbf{Y}^{(L-1)}\boldsymbol{\Theta}^{(L)}$$

  - ⇒ $\mathbf{U}^{(\ell)}$ increases size of intermediate signals $\mathbf{Y}^{(\ell-1)}$ since $N^{(0)} < N$

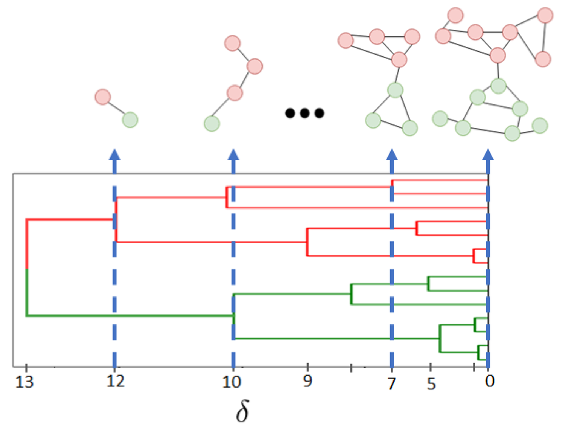


Features of the architecture

- The graph topology is incorporated via the **clustering-based** design of $\mathbf{U}^{(\ell)}$
- The reduced dimensionality of $\mathbf{Z}$ implicitly limits the **degrees of freedom**
  - ⇒ The GDec is more robust to noise but more sensitive to model mismatch

Upsampling Operators for GDec

- Designing an upsampling operator is straightforward for regular signals
  - ⇒ But is a non-trivial task when dealing with graph signals
- **Our solution:** rely on agglomerative hierarchical clustering to obtain a dendrogram
  - ⇒ Cutting at $L + 1$ resolutions to obtain a collection of node sets [Day84]



- **Parent-child relations** from the dendrogram encoded in $\mathbf{P}^{(\ell)} \in \{0, 1\}^{N^{(\ell)} \times N^{(\ell-1)}}$
  - ⇒ $P_{ij}^{(\ell)} = 1$ if node i in layer ℓ is the child of node j in layer $\ell - 1$
- Define the upsampling operator as the convex combination

$$\mathbf{U}^{(\ell)} = (\gamma\mathbf{I} + (1 - \gamma)\mathbf{A}^{(\ell)})\mathbf{P}^{(\ell)} = \tilde{\mathbf{H}}^{(\ell)}\mathbf{P}^{(\ell)}$$

  - ⇒ $\mathbf{A}^{(\ell)} \neq 0$ defined based on known $\mathbf{A}$

Analysis of the denoising capability

- Consider a simplified 2-layer GCG denoted as $f_{\boldsymbol{\Theta}}(\mathbf{H})$
  - ⇒ With expected square Jacobian of $f_{\boldsymbol{\Theta}}(\mathbf{H})$ diagonalized as $\boldsymbol{\mathcal{X}} = \mathbf{W}\boldsymbol{\Sigma}\mathbf{W}^\top$
  - ⇒ Assuming that $\mathbf{x}_0$ is a **bandlimited** graph signal and $\mathcal{G}$ is drawn from SBM
- **Key:** establishing a relation between the K leading eigenvectors $\mathbf{V}_K$ and $\mathbf{W}_K$
  - ⇒ Feasible through the expectations $\boldsymbol{\mathcal{A}} = \mathbb{E}[\mathbf{A}]$ and $\bar{\boldsymbol{\mathcal{X}}} = \mathbb{E}[\boldsymbol{\mathcal{X}}]$

$$\mathbf{A} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top \iff \boldsymbol{\mathcal{A}} = \bar{\mathbf{V}}\bar{\boldsymbol{\Lambda}}\bar{\mathbf{V}}^\top$$
$$\Updownarrow$$
$$\boldsymbol{\mathcal{X}} = \mathbf{W}\boldsymbol{\Sigma}\mathbf{W}^\top \iff \bar{\boldsymbol{\mathcal{X}}} = \bar{\mathbf{W}}\bar{\boldsymbol{\Sigma}}\bar{\mathbf{W}}^\top$$

Theorem

Let $\mathbf{x}_0$ be a K-bandlimited graph signal spanned by $\mathbf{V}_K$. For $N > N_{\epsilon,\delta}$, the error for each iteration t of SGD with stepsize η is bounded as

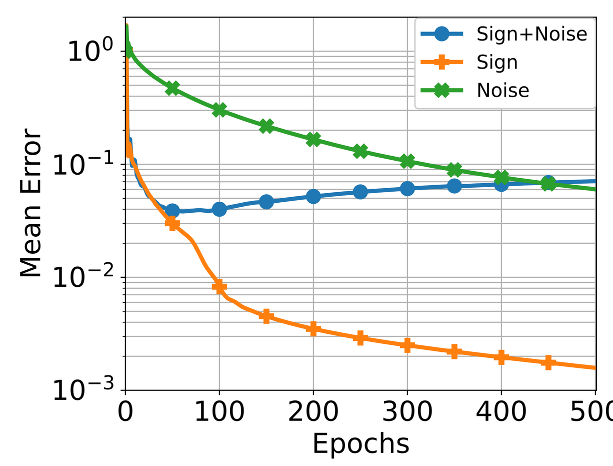
$$\|\mathbf{x}_0 - f_{\boldsymbol{\Theta}_{(t)}}(\mathbf{H})\|_2 \le \left((1 - \eta\sigma_K^2)^t + \delta(1 - \eta\sigma_N^2)^t\right)\|\mathbf{x}_0\|_2$$
$$+ \xi\|\mathbf{x}\|_2 + \sqrt{\sum_{i=1}^{N}((1 - \eta\sigma_i^2)^t - 1)^2(\mathbf{w}_i^\top\mathbf{n})^2},$$

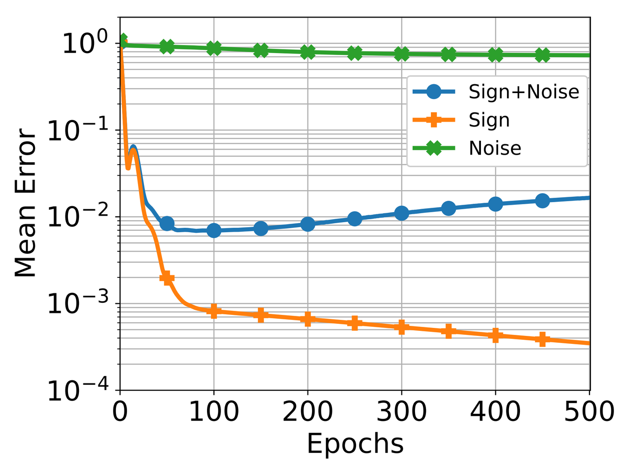
with probability at least $1 - e^{-F^2} - \phi - \epsilon$.

- The first term models the signal error and the eigenvector misalignment
- The third term captures the error resulting from learning the noise
  - ⇒ $\mathbf{x}_0$ is learned faster than noise so the error decreases for the first iterations
  - ⇒ If too many iterations are considered the noise is learned and the error increases

Numerical results: Synthetic data

- Graphs are SBM with $N = 64$ nodes and $K = 4$ communities
  - ⇒ "Signal" ($\mathbf{x}_0$) is a piece-wise constant signal
- The results show that the signal $\mathbf{x}_0$ is fitted faster than the noise $\mathbf{n}$
  - ⇒ The best error is achieved after a few iterations
  - ⇒ The GDec fits the noise much more slowly than the GCG





Numerical results: Real data

- We test the performance of the proposed GNNs in a wide range of settings
  - ⇒ Using temperature data, financial data from S&P 500, and the Cora dataset
  - ⇒ Considering Gaussian, Uniform, or Bernoulli noise
- Compare the performance with several convex and non-linear models

DATASET (METRIC)	METHOD	BL	TV	LR	GTF	MED	GCNN	GAT	K-GAE	GUSC	GCG	GDec
TEMPERATURE (NMSE)	Gaussian	0.062	0.117	0.095	0.066	0.053	0.123	0.045	0.134	0.044	0.056	**0.035**
	Uniform	0.063	0.117	0.094	0.064	0.053	0.118	0.047	0.136	0.049	0.057	**0.036**
S&P 500 (NMSE)	Gaussian	0.350	0.238	0.231	0.239	0.319	0.252	0.199	0.354	0.203	**0.188**	**0.188**
	Uniform	0.216	0.246	0.161	0.298	0.340	**0.091**	0.222	0.273	0.127	**0.094**	0.121
CORA (ERROR RATE)	Whole $\mathcal{G}$	0.154	0.142	0.115	0.126	0.167	0.099	0.141	0.135	0.099	**0.093**	0.121
	Conn. comp.	0.151	0.141	0.105	0.116	0.165	0.093	0.139	0.135	0.094	**0.088**	0.125

- The GCG and/or GDec outperform the alternatives in most settings
  - ⇒ GDec outperforms the alternatives in the temperature dataset with smooth signals