SCCGNN: any ideas?

Anton Savostianov[1] • **Francesco Tudisco**[1] • **Nicola Guglielmi**[1]
[1]Gran Sasso Science Institute, viale F.Crispi 7, L'Aquila, Italy, 67100, email: anton.savostianov@gssi.it

Abstract: Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Keywords: NNs

I. Things

Let $\mathcal{K}$ be a simplicial complex where $\mathcal{V}_k(\mathcal{K})$ is the set of k-order simplices in $\mathcal{K}$, $\mathcal{K} = \mathcal{V}_0(\mathcal{K}) \cup \mathcal{V}_1(\mathcal{K}) \cup \mathcal{V}_2(\mathcal{K}) \cup \ldots$; let $m_k = |\mathcal{V}_k(\mathcal{K})|$. Let $B_k \in \mathrm{Mat}_{m_{k-1}\times m_k}$ be a boundary operator mapping simplex of order k to its border of the order $k-1$; by the fundamental lemma of topology $B_k B_{k+1} = 0$.

Given the lemma above, the homology group $\mathcal{H}_k = {}^{\operatorname{im} B_k}\!/\!_{\ker B_{k+1}}$ is correctly defined; the elements of $\mathcal{H}_k$ correspond to k-dimensional holes in the simplicial complex $\mathcal{K}$. Through the harmonic representative it is convenient to exploit the isomorphism:

$$\mathcal{H}_k \cong \ker\left(B_k^\top B_k + B_{k+1}B_{k+1}^\top\right)$$

Operators $L_k^\downarrow = B_k^\top B_k$, $L_k^\uparrow = B_{k+1}B_{k+1}^\top$ and $L_k = L_k^\downarrow + L_k^\uparrow$ are referred as $k-$th order up-, down- and complete graph Laplacians.

Algebra of boundary operators admits an important full space decomposition:

Lemma 1 **(Hodge Decomposition, [Lim19, Thm. 5.2])** For the k-th order Hodge Laplacian $L_k \in \mathbb{R}^{m_k\times m_k}$, the following decomposition holds:

$$\mathbb{R}^{m_k} = \overbrace{\operatorname{im} B_k^\top \oplus \ker L_k}^{\ker B_{k+1}^\top} \oplus \operatorname{im} B_{k+1}$$

$$\text{with } \underbrace{\ker L_k \oplus \operatorname{im} B_{k+1}}_{\ker B_k}$$

We also assume the case of weighted simplicial complex; in such situations the family of k weight functions are introduced: $w_k(\cdot) : \mathcal{V}_k(\mathcal{K}) \mapsto (0; +\infty)$ with corresponding diagonal weight matrices $W_k \in \mathrm{Mat}_{m_k\times m_k}$ where $[W_k]_{ii} = w_k(\sigma_i)$, $\sigma_i \in \mathcal{V}_k(\mathcal{K})$. Then the weighting scheme preserving the homology definition can be designed as follows:

$$B_k \to W_{k-1}^{-1} B_k W_k$$

Weighting does not change the dimensionality of the homology group and the remaining terms in the decomposition; isomorhism and the decomposition above hold in the weighted case as well.

here we need a proper discussion of the normalisation properties in the general case

II. SCCGNN

Graph Fourier Transform. Let L be a graph Laplacian (of any order); then L_k is simmetric positive semidefinite and has a complete set of orthonormal eigenvectors $U = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots \boldsymbol{u}_n]$, $L_k = U\Lambda U^\top$. Then graph Fourier transform of a simplicial signal $\boldsymbol{x} \in C_k$ is defined as $\hat{\boldsymbol{x}} = U^\top \boldsymbol{x}$ and the convolution on graphs:

$$\boldsymbol{x} \star \boldsymbol{y} = U\left(\left(U^\top x\right) \odot \left(U^\top \boldsymbol{y}\right)\right) = U \operatorname{diag}\left(U^\top x\right) U^\top \boldsymbol{y} \qquad \text{(Eqn. 1)}$$

exactly the same as in the discrete Fourier transform with a different basis instead of $e^{-2\pi i k/n}$

Then graph filter g_θ act as $g_\theta(L_k) = U g_\theta(\Lambda) U^\top$; since $g_\theta(\Lambda)$ is diagonal, every filter application $g_\theta(L_k)\boldsymbol{x}$ is a graph convolution. Hence, for each convolution of $\boldsymbol{x}$ with a kernel $\boldsymbol{w}$, one can find a graph filter

$$\operatorname{diag}\left(U^\top \boldsymbol{w}\right) = g_\theta(\Lambda) \qquad \text{(Eqn. 2)}$$

Instead of the unpleasant and computationally demanding generic case of $g_\theta(L_k)$, one can use approximations by the polynomial filters of a smaller degree:

T_i are Chebyshev polynomials

$$g_\theta(L_k) \approx \sum_{i=0}^{K} \theta_i T_i(\hat{L}_k) \approx \sum_{i=0}^{K} \alpha_i L_k^i \qquad \text{(Eqn. 3)}$$

Convolutional Layers. Let $\boldsymbol{x}_i \in C_1$; then a convolutional layer induced by the simplicial geometry is given by:

$$\boldsymbol{x}_{n+1} = \sigma\left(\sum_{i=0}^{K} \alpha_i^{(n)} L_1^i \boldsymbol{x}_n\right), \qquad \text{where } \sigma = \text{reLU and } \alpha_i^{(n)} \in \mathbb{R}$$

where $\sigma(x) = x \cdot \chi(x)$. Note that $L_1^i = \left(B_1^\top B_1\right)^i + \left(B_2 B_2^\top\right)^i = L_1^{\downarrow i} + L_1^{\uparrow i}$; let $P_K(\boldsymbol{\alpha}, A) = \sum_{i=0}^{K} \alpha_i A^i$ be a matrix polynomial. Then, the convolution layer can be rewritten as

$$\boldsymbol{x}_{n+1} = \sigma\left(\sum_{i=0}^{K} \alpha_i^{(n)} L_1^i \boldsymbol{x}_n\right) = \sigma\left(P_K(\boldsymbol{\alpha}^{(n)}, L_1)\boldsymbol{x}_n\right) = \sigma\left(P_K(\boldsymbol{\alpha}^{(n)}, L_1^{\downarrow})\boldsymbol{x}_n + P_K(\boldsymbol{\alpha}^{(n)}, L_1^{\uparrow})\boldsymbol{x}_n\right) \qquad \text{(Eqn. 4)}$$

Equation above implies the same vector of polynomial coefficients for both up- and down-terms. In a more general case, one could consider a layer with two independent polynomials:

$$\boldsymbol{x}_{n+1} = \sigma\left(P_K(\boldsymbol{\alpha}^{(n)}, L_1^{\downarrow})\boldsymbol{x}_n + Q_K(\boldsymbol{\beta}^{(n)}, L_1^{\uparrow})\boldsymbol{x}_n\right) \qquad \text{(Eqn. 5)}$$

Let us omit $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in the polynomial arguments for clarity (we will refer to the coefficients of P_K as α_i and Q_k as β_i respectively).

Assuming one can extend the dimension of the input for the hidden layers, $\boldsymbol{x}_n \in \mathbb{R}^{m_1 \times 1}$ and $\boldsymbol{x}_{n+1} \in \mathbb{R}^{m_1 \times H}$ and $\alpha_i^{(n)}, \beta_i^{(n)} \in \mathbb{R}^{1 \times H}$.

the order of mutliplication changes, $P_K(\boldsymbol{\alpha}^{(n)}, L_1)\boldsymbol{x}_n = \sum_{i=0}^{K} L_1^i \boldsymbol{x}_n \alpha_i^{(n)}$

Training, Batching, Masking and the Task. Let $\boldsymbol{x} \in C_1$ be a given given flow on edges; we separate vector's entries into two parts — missing and complete, $\boldsymbol{x} = [\boldsymbol{x}_c; \boldsymbol{x}_m]$; the missing part of the vector is missing and is unavailible during training. Instead, we assume that the vector $\boldsymbol{x}$ that arrives as the input of SCCGNN is $\boldsymbol{x} = [\boldsymbol{x}_c; \boldsymbol{x}_m]$ where $\boldsymbol{x}_m = \phi$ where ϕ is a filler value for the missing entries, initial guess. One may opt for $\phi = 0$ or $\phi = \text{median}\,(\boldsymbol{x}_c)$.

this is going to be a little shocking but bear with me

Problem: Train a SCCGNN $f(\boldsymbol{x})$ composed of convolutional graph layers such that

$$\| f(\boldsymbol{x}) \mid_m - \boldsymbol{x}_m \| \to \min \qquad \text{(Eqn. 6)}$$

without exploiting values from $\boldsymbol{x}_m$.

Remark: (Batching and SGD without S) By the nature of the task $\boldsymbol{x}$ maybe the only data point available; this implies that data points cannot be grouped into batches and the optimisation with SGD loses the stochasticity.

Naturally, one may try to sample entries of $\boldsymbol{x}$ to form a batch, but this brute force idea contradicts the interconnected nature of the graph neural network. One can find a substantial discussion on the proper subsampling for GNNs (which normally mean subsampling vertices with their neighbours via specifically constructed distribution, in other words), but we propose the alternative batching and randomisation.

Since the network has the access only to the known entries $\boldsymbol{x}_c$, it should be train to maintain correct values on the known entries (e.g. by minimizing $\|f(\boldsymbol{x})\mid_c - \boldsymbol{x}_c\|$) which has a global (but maybe not unique) optimum at $f(\boldsymbol{x}) = \boldsymbol{x}$ (identity). In order to avoid the trivial and incorrect sollution one may propose the following masking idea:

- ◇ let $\boldsymbol{e} = [\boldsymbol{e}_c; \boldsymbol{e}_m]$ be a binary vector such that $\boldsymbol{e}_m = 0$ and each entry of in the complete part be $\sim \text{Bernoulli}(p)$;
- ◇ then $\boldsymbol{x} \odot \boldsymbol{e}$ is a subsampling where we "forget" about some of the known entries;
- ◇ let us train the network with a loss:

$$\|f(x \odot \boldsymbol{e})\mid_c - \boldsymbol{x}_c\|^2 \qquad \text{(Eqn. 7)}$$

  so it avoids the identity and tries to keep the original known values intact;
- ◇ then the epoch is naturally formed by $1/p$ steps (such that each entry of $\boldsymbol{x}_c$ enters the training).

II. References

[EDS20] Stefania Ebli, Michaël Defferrard, and Gard Spreemann. Simplicial Neural Networks, December 2020.

[Lim19] Lek-Heng Lim. Hodge Laplacians on graphs, August 2019.

[YI23] Maosheng Yang and Elvin Isufi. Convolutional Learning on Simplicial Complexes, January 2023.