Understanding Inductive Biases of Graph Neural Networks

Kimon Fountoulakis, Edward Poon @CS UWaterloo, Andreas Loukas @EPFL
22/06/2020

SIAM MDS20
Graph neural network goals

- Combine graph data with node and edge features to solve prediction problems on graphs
  - Node classification
  - Link prediction
  - Graph classification
  - Learning combinatorial problems
Our goals

- Understand inductive biases
  - Do graph neural networks have inherent biases in solving certain problems?
Graph neural networks - message passing architectures

-Multiple architectures:
  - ChebConv, SAGEConv, **GraphConv**, GatedGraphConv, **GATConv**, AGNNConv, TAGConv, **GINConv**, ARMAConv, SignedConv, PointConv, NNConv and the list goes on…

- **GraphConv**: graph convolution
- **GATConv**: puts attention on edges by learning the edge weights
- **GINConv**: introduces skip connections
Message passing architecture

\[ \text{output features for each node} = \text{activation(aggregation( input features for each node ))} \]

- The activation and aggregation functions depend on the architecture.
Example: GraphConv or GCN (Graph Convolutional Network)

$$X_k = \text{activation}(D^{-1}AX_{k-1}W_k)$$

- $A$: adjacency matrix
- $D$: degree matrix
- activation: ReLU, Tanh etc
- Note: I ignored the intercept for simplicity
Assumptions

-The last layer output $x_d$ maps nodes to $[-1, 1]$ (Binary Classification)

-The final label $y$ is set to $[y]_i = -1$ if mapping $[x_d]_i < 0$, and $[y]_i = 1$ if $[x_d]_i \geq 0$

-The output of each layer is bounded $\|D^{1/2}X_k\|_2 \leq \beta \ \forall k$
Definitions

Definition [Confidence]: \[ \sum_{i=1}^{n} (1 - |[x_d]_i|)^2 \]

where \( n \) is the number of nodes and \( x_d \) is the output mapping of length \( n \).

Definition [\( \gamma \)-margin classifier]: A classifier with confidence less than \( \gamma \).
Cut upper bound for binary classification of nodes

Let $(S_0, S_1)$ be a bi-partition of the graph such that $S_1 = \{ i \in V : [y]_i = 1 \}$ and $S_{-1} = S_0^c$

Let $\text{cut}(S_1, S_{-1}) = \text{total sum of weights of edges among } S_1 \text{ and } S_{-1}$.

Let $\mathcal{L}$ be the normalized Laplacian $D^{-1/2}LD^{-1/2}$, where $L = D - A$ is the Laplacian matrix.
Cut upper bound for binary classification of nodes

- Then, 

$$\text{cut}(S_1, S_{-1}) \leq \frac{1}{4} \left( \sqrt{\gamma \| \mathcal{L} \|_2} + \left( \prod_{l=1}^{d} \| W_l \|_2^{c \frac{d-l}{d}} \right)^{\xi \frac{c}{2} \alpha \frac{1}{2 - 1 - c^{1/d}}} \right)^2$$

- Bound on confidence of the model
- Bound on output of each layer
- $\| D^{1/2} X_k \|_2 \leq \beta$

$c$ is a constant chosen in $[0,1]$. The product of norms of weights, might increase with as a function of the number of layers $d$. 

- $\beta$
Cut upper bound for binary classification of nodes

Then,

$$\text{cut}(S_1, S_{-1}) \leq \frac{1}{4} \left( \sqrt{\gamma \| L \|_2} + \left( \prod_{l=1}^{d} \| W_l \|_2^{c \frac{d-l}{d}} \right)^{\frac{1}{2}} \right)^2$$

- Norm of normalized Laplacian $L$

- $\alpha$ is a function of $L$ which satisfies $\alpha(L) \in (0,1)$.

The quantity in the box is decreasing monotonically as a function of the number of layers $d$.

$$\alpha = \max_{\nu \in \text{eig. val. of } L} \nu^{1-c^{1/d}} (1 - \nu)^2,$$ where $p \in [0,1]$.
Cut upper bound for binary classification of nodes

-Then,

$$\text{cut}(S_1, S_{-1}) \leq \frac{1}{4} \left( \sqrt{\gamma \| \mathcal{L} \|_2} + \left( \prod_{l=1}^{d} \| W_l \|_2 \right)^{\frac{d-l}{d}} \xi \frac{c}{2} \alpha \frac{1}{1 - c^{1/d}} \beta^{1-c} \right)^2$$

$$\xi = \text{trace}(X_0^T \mathcal{L} X_0)$$, measures the smoothness of the input data $X_0$
Cut upper bound for binary classification of nodes

Then,

$$\text{cut}(S_1, S_{-1}) \leq \frac{1}{4} \left( \sqrt{\| \mathcal{L} \|_2} + \left( \prod_{l=1}^{d} \| W_l \|_2^{\frac{d-l}{d}} \right) \right)^2$$

- **Red boxes:** GCN related terms
- **Green boxes:** Graph related terms
- **Blue box:** Smoothness of features
Extensions of the bound

-The bound extends to other convolutional networks of the form

\[ X_k = \text{activation}(PX_{k-1}W_k) \]

-where \( P \) is a symmetric matrix with same eigenvector as \( \mathcal{L} \)

-The bound extends to normalized convolutional networks.

-The bound does not extend to graph convolutional networks with skip connections.
Corollary for planted clique graphs

- Consider a planted clique graph where
  - Graph has \( n \) nodes
  - \( k \) out of \( n \) consist a clique
  - All pairs of nodes in the clique are connected with an edge
  - Any other pair of nodes is connected with probability \( 0 < q < 1 \)

- GCN will not cut a clique with size that satisfies

\[
k > \frac{1}{4} \left( \sqrt{\gamma \|L\|_2} + \left( \prod_{l=1}^{d} \|W_l\|_2 \right)^{\frac{d-l}{d}} \xi \gamma^2 \alpha^2 \frac{1-\alpha}{1-c/1/d} \beta^{1-c} \right)^2
\]
Will GCN cut the clique in practice? Dataset details

- Each dataset has 15 graphs of varying size from 50 to 200 nodes.

- \( q = 0.1 \)

- Ratio \( \frac{100k}{n} \) (clique percentage) is fixed for each dataset

- We use 60% of the labels of a graph for training. All labels of the clique are included in the training set

- 40% of the training nodes in the clique are on purpose mis-labelled.
Will GCN cut the clique in practice? Labels in the training dataset

- Normally, nodes in the clique that are used for training would be given the same label, 1 or -1.

- This is because, usually the goal for the planted clique problem is to detect the clique.

- Our goal is to force the model the cut the clique, i.e., the predicted labels of nodes in clique have both 1 and -1 labels.

- Therefore, 40% of labels of nodes in the clique are on purpose given label 1 and the remaining 60% are given label -1.
Will GCN cut the clique in practice? Tuning details

- Input features for each node is its degree ($X_0 = degrees$) or its indicator vector ($X_0 = I_n$). We got similar results for both settings.

- Each layer has width 50, i.e., $W_k \in \mathbb{R}^{h_{k-1} \times 50}$, where $h_0 = 1$ for the first layer, $h_{k-1} = 50$ for $k > 1$, and $W_d \in \mathbb{R}^{50 \times 1}$ for the last layer $d$.

- For each setting we average over 5 trials (random weights).

- We train for 1000 epochs, with decaying learning rate.

- Hinge loss was used.
Will GCN cut the clique in practice? No.

Blocks show the percentage of cliques that are cut.
Is GCN confident? Yes.

-Blocks show confidence level scaled from 0 to 1.
Does GCN fit the data? No

-Blocks show minimum loss function over 5 random initial weights.
Varying incorrect clique labels in the training dataset

- One graph of 100 nodes was used as data
- Minimum loss is reported over 10 trials (initial weights)
Optimization landscape for planted clique graphs

40% mixed training labels in the clique

0% mixed training labels in the clique

Flat region, where all nodes are put in one class

-The origin is the local minimum

-Dimensionality reduction has been used
Thank You!