Predict Then Propagate

PREDICT THEN PROPAGATE: GRAPH NEURAL NETWORKS MEET PERSONALIZED PAGERANK

By The Penultimate BOZ
(Daniel Borhegyi, Henry Odza and Mike Zhang)

Main Idea/Problem of The Paper

- Understanding and predicting graphs is useful
- Graph Convolutional Networks (GCNs) are limited because of overfitting for aggregation for further neighbors
- Solutions to this before this paper are more time consuming and only a bit more accurate
- The solutions of PPNP and APPNP proposed here are more accurate and only a bit more time consuming than GCN
    - A core component of this is implementing Page Rank

Page Rank

Original PageRank Function:

$$\boldsymbol{\pi}_{\text{pr}} = \boldsymbol{A}_{\text{rw}}\boldsymbol{\pi}_{\text{pr}},$$
$$\boldsymbol{A}_{\text{rw}} = \boldsymbol{A}\boldsymbol{D}^{-1}.$$

Personalized Page Rank

$$\boldsymbol{\pi}_{\text{ppr}}(\boldsymbol{i}_x) = (1-\alpha)\hat{\boldsymbol{A}}\boldsymbol{\pi}_{\text{ppr}}(\boldsymbol{i}_x)+\alpha\boldsymbol{i}_x,$$
$$\boldsymbol{\pi}_{\text{ppr}}(\boldsymbol{i}_x) = \alpha\left(\boldsymbol{I}_n - (1-\alpha)\hat{\boldsymbol{A}}\right)^{-1}\boldsymbol{i}_x.$$

Fully Personalized PageRank Matrix

$$\boldsymbol{\Pi}_{\text{ppr}} = \alpha(\boldsymbol{I}_n - (1-\alpha)\hat{\boldsymbol{A}})^{-1},$$

A_{rw}: The row-normalized adjacency matrix.

A: The adjacency matrix of the graph, where A_{ij}=1 if there is an edge between nodes i and j, and A_{ij}=0 otherwise.

D^{-1}: The inverse of the degree matrix D, a diagonal matrix where D_iD_i is the degree (number of neighbors) of node i. Normalizing by D^{-1} ensures each row of A_{rw} sums to 1

A: The symmetrically normalized adjacency matrix:

$$\hat{A} = D^{-1/2}AD^{-1/2}$$

α: The teleport (or restart) probability. It determines how likely the random walk is to "teleport" back to the root node x at each step.

i_x: The teleport vector. It is a one-hot vector where the entry for node x is 1, and all other entries are 0. It specifies the root node for personalization.

Proof of Existence

$$\mathbf{\Pi}_{\text{ppr}} = \alpha \left(\boldsymbol{I}_n - (1-\alpha)\hat{\tilde{\boldsymbol{A}}} \right)^{-1}$$

In: The identity matrix of size n×n, where n is the number of nodes in the graph. The identity matrix ensures that teleportation retains focus on the starting node.

exists iff the determinant $\det(\boldsymbol{I}_n - (1-\alpha)\hat{\tilde{\boldsymbol{A}}}) \neq 0$, which is the case iff $\det(\hat{\tilde{\boldsymbol{A}}} - \frac{1}{1-\alpha}\boldsymbol{I}_n) \neq 0$, i.e. iff $\frac{1}{1-\alpha}$ is not an eigenvalue of $\hat{\tilde{\boldsymbol{A}}}$. This value is always larger than 1 since the teleport probability $\alpha \in (0, 1]$. Furthermore, the symmetrically normalized matrix $\hat{\tilde{\boldsymbol{A}}}$ has the same eigenvalues as the row-stochastic matrix $\tilde{\boldsymbol{A}}_{\text{rw}}$. This can be shown by multiplying the eigenvalue equation $\hat{\tilde{\boldsymbol{A}}}\boldsymbol{v} = \lambda\boldsymbol{v}$ with $\tilde{\boldsymbol{D}}^{-1/2}$ from left and substituting $\boldsymbol{w} = \tilde{\boldsymbol{D}}^{-1/2}\boldsymbol{v}$. This also shows that the eigenvectors of $\hat{\tilde{\boldsymbol{A}}}$ are the eigenvectors of $\tilde{\boldsymbol{A}}_{\text{rw}}$ scaled by $\tilde{\boldsymbol{D}}^{1/2}$. The largest eigenvalue of a row-stochastic matrix is 1, as can be proven using the Gershgorin circle theorem. Hence, $\frac{1}{1-\alpha}$ cannot be an eigenvalue and $\mathbf{\Pi}_{\text{ppr}}$ always exists.

PPNP Equation - Personalized Propagation of Neural Predictions

Modified page rank to include to include root node with ***teleport vector***

$$\boldsymbol{\pi}_{\text{ppr}}(\boldsymbol{i}_x) = \alpha \left(\boldsymbol{I}_n - (1-\alpha)\hat{\tilde{\boldsymbol{A}}}\right)^{-1} \boldsymbol{i}_x.$$

$$\boldsymbol{Z}_{\text{PPNP}} = \text{softmax}\left(\alpha \left(\boldsymbol{I}_n - (1-\alpha)\hat{\tilde{\boldsymbol{A}}}\right)^{-1} \boldsymbol{H}\right), \qquad \boldsymbol{H}_{i,:} = f_\theta(\boldsymbol{X}_{i,:}),$$

$$s\left(x_i\right) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

softmax is a common machine learning function that normalizes a vector into numbers between 0 and 1 that add up to 1 (a probability distribution!)

$\boldsymbol{i_x}$ becomes ***H*** because all the nodes are generalized into a neural network which makes predictions based on x then within the rest of the equation it is smoothed across the network

APPNP Equation

Using the power iteration method.

$$
\begin{aligned}
\boldsymbol{Z}^{(0)} &= \boldsymbol{H} = f_\theta(\boldsymbol{X}), \\
\boldsymbol{Z}^{(k+1)} &= (1-\alpha)\hat{\tilde{\boldsymbol{A}}}\boldsymbol{Z}^{(k)} + \alpha\boldsymbol{H}, \\
\boldsymbol{Z}^{(K)} &= \operatorname{softmax}\left((1-\alpha)\hat{\tilde{\boldsymbol{A}}}\boldsymbol{Z}^{(K-1)} + \alpha\boldsymbol{H}\right),
\end{aligned}
$$

H is the initial prediction matrix (output of the neural network applied to the node features). A^ is the symmetrically normalized adjacency matrix with self-loops.

α (Teleport Probability):

Controls how much of the initial prediction (H) is retained in each propagation step.

k : number of iterations.

Convergence of APPNP

$$\boldsymbol{Z}^{(k+1)} = (1-\alpha)\hat{\tilde{\boldsymbol{A}}}\boldsymbol{Z}^{(k)} + \alpha\boldsymbol{H}. \tag{6}$$

After the k-th propagation step, the resulting predictions are

$$\boldsymbol{Z}^{(k)} = \left((1-\alpha)^k \hat{\tilde{\boldsymbol{A}}}^k + \alpha \sum_{i=0}^{k-1} (1-\alpha)^i \hat{\tilde{\boldsymbol{A}}}^i\right) \boldsymbol{H}. \tag{7}$$

If we take the limit $k \to \infty$ the left term tends to 0 and the right term becomes a geometric series. The series converges since $\alpha \in (0, 1]$ and $\hat{\tilde{\boldsymbol{A}}}$ is symmetrically normalized and therefore $\det(\hat{\tilde{\boldsymbol{A}}}) \leq 1$, resulting in

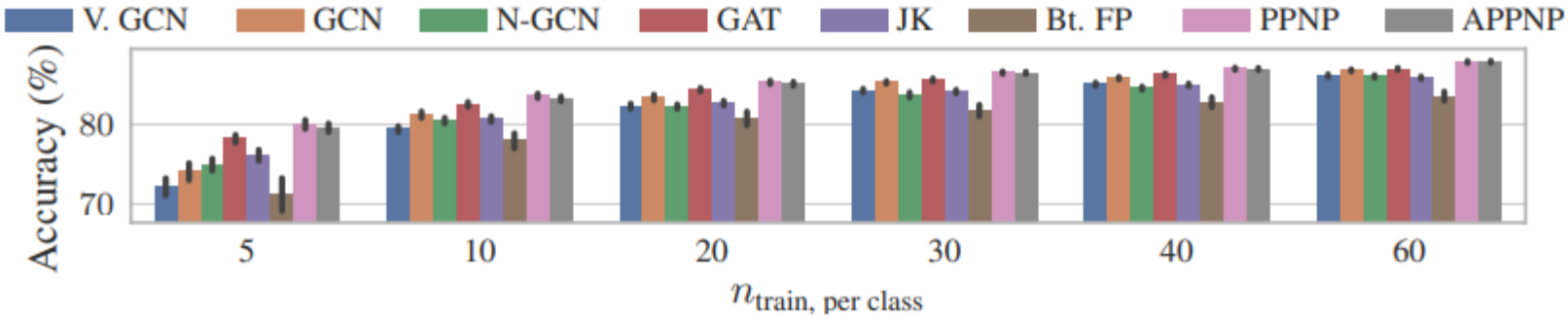
$$\boldsymbol{Z}^{(\infty)} = \alpha \left(\boldsymbol{I}_n - (1-\alpha)\hat{\tilde{\boldsymbol{A}}}\right)^{-1} \boldsymbol{H}, \tag{8}$$

Compared to Prior Solutions

Table 3: Average training time per epoch. PPNP and APPNP are only slightly slower than GCN and much faster than more sophisticated methods like GAT.

Graph	V. GCN	GCN	N-GCN	GAT	JK	Bt. FP*	PPNP**	APPNP
CITESEER	37.6 ms	35.3 ms	115.9 ms	187.0 ms	57.5 ms	-	49.2 ms	43.3 ms
CORA-ML	32.4 ms	36.5 ms	118.9 ms	217.4 ms	43.6 ms	-	55.3 ms	42.7 ms
PUBMED	48.6 ms	48.3 ms	342.6 ms	1029.8 ms	77.8 ms	-	-	64.1 ms
MS ACADEMIC	45.5 ms	39.2 ms	328.5 ms	772.2 ms	61.9 ms	-	-	59.8 ms

*not applicable, since core method not trainable **out of memory on PUBMED, MS ACADEMIC (see efficiency analysis in Section 3)



Our Project Based on This

$$\boldsymbol{Z}_{\text{PPNP}} = \text{softmax}\left(\alpha\left(\boldsymbol{I}_n - (1-\alpha)\hat{\tilde{\boldsymbol{A}}}\right)^{-1}\boldsymbol{H}\right), \qquad \boldsymbol{H}_{i,:} = f_\theta(\boldsymbol{X}_{i,:}),$$

- Use APPNP function
- Generate features from classes in homework data by applying random noise on that data
  - Fake Train H function
- Make the necessary variables, apply the algorithm
- Test on Test Data with more and more noise to get a graph of accuracy

α = .5

$\boldsymbol{H}$ = dataset in one hot representation with noise

$\boldsymbol{I_n}$ = identity matrix

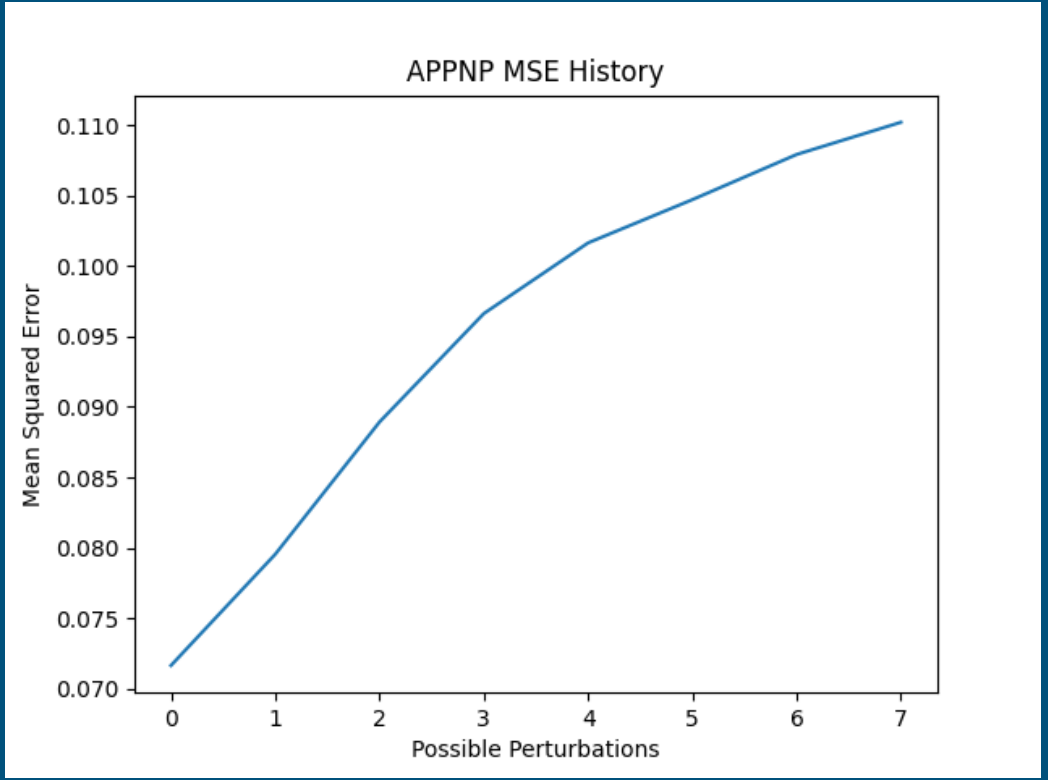
$\hat{\tilde{\boldsymbol{A}}} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$ =

Problems with APPNP

- APPNP is heavily influenced by H
- H is calculated by a neural network matching features to classes
- What happens when there is noise in the feature data?

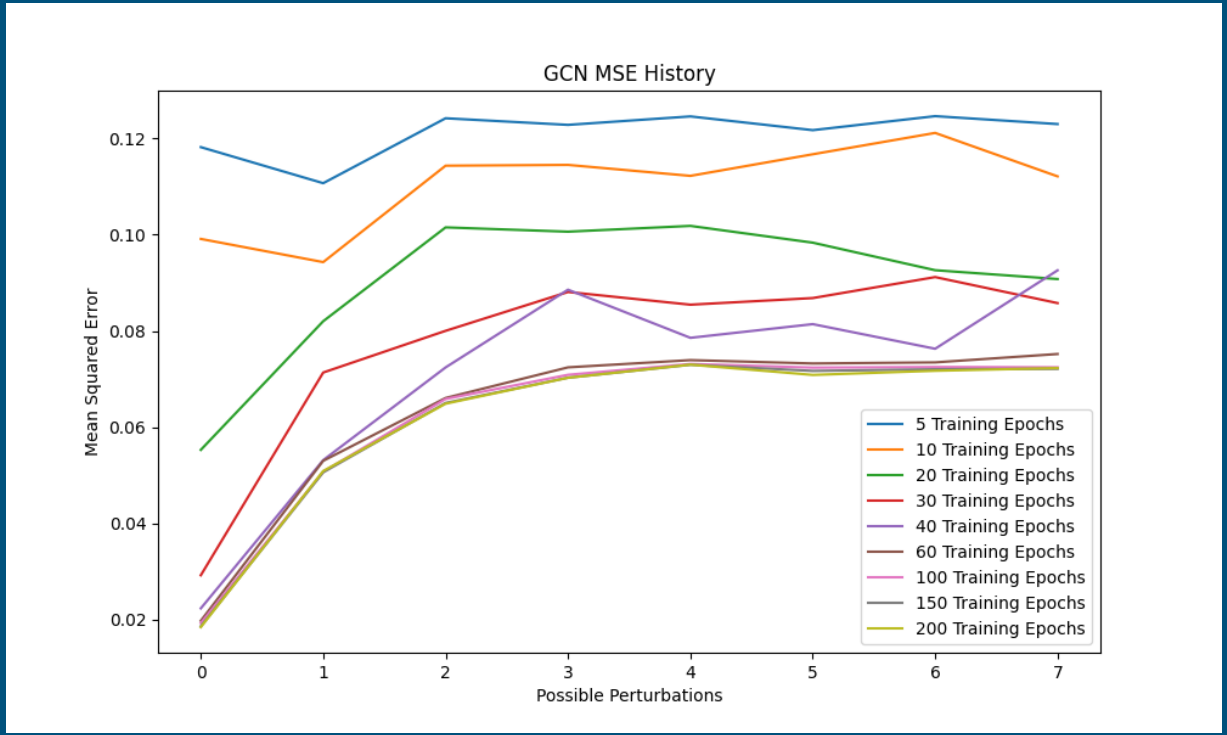
Results 1

Simulated noise on HW3 graph and its effect on accuracy

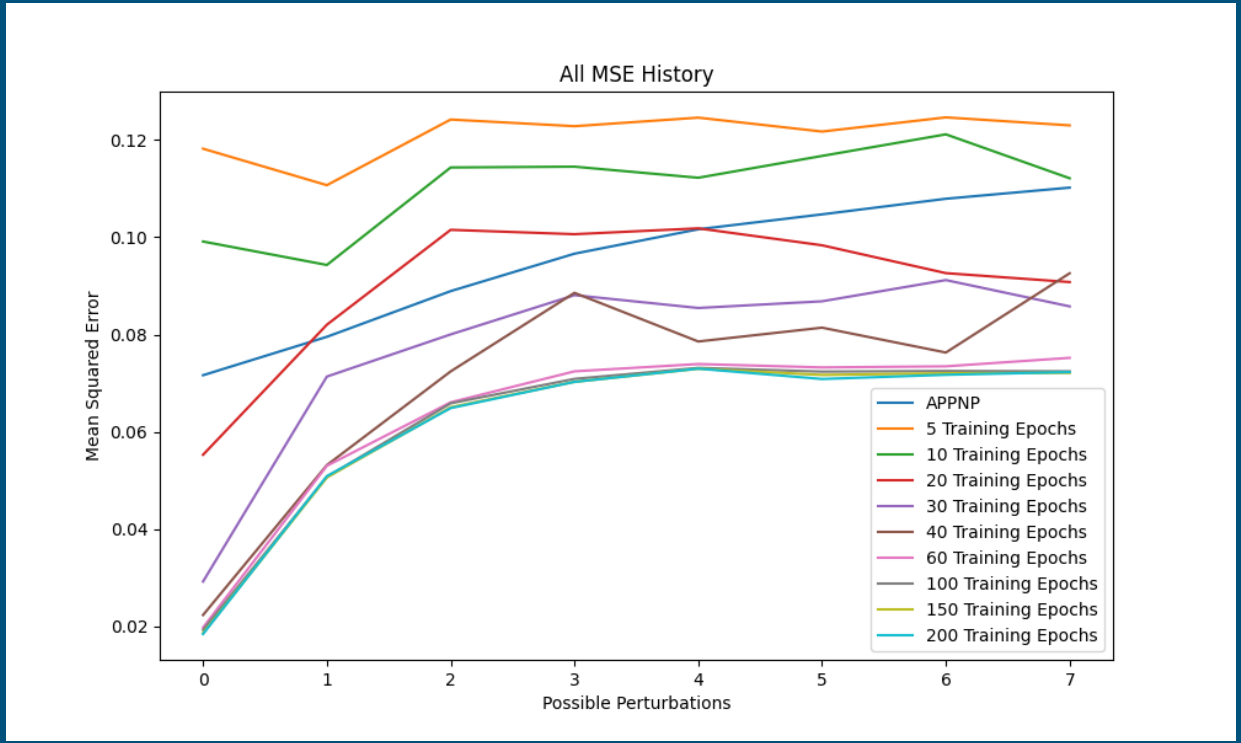


Results 2

Simulated noise on simulated features and its effect on accuracy



Results 3


All MSE History
Mean Squared Error
0.12
0.10
0.08
0.06
0.04
0.02
0
1
2
3
4
5
6
7
Possible Perturbations
APPNP
5 Training Epochs
10 Training Epochs
20 Training Epochs
30 Training Epochs
40 Training Epochs
60 Training Epochs
100 Training Epochs
150 Training Epochs
200 Training Epochs

Questions?