BCB726

Lecture 5

November 3, 2025

Today

- Introduction to graph neural networks (GNNs)
- Network neural networks vs label propagation
- Studying spatial omics data with GNNs

Graph representations of data

- A graph is a collection of **nodes** and **edges** that can be used to represent relationships (edges) between a set of entities (nodes).
- Edges may be spatial (e.g. proximal), correlation (e.g. similar), or physical (e.g. binding)
- Examples of graph structures we might encounter :
  - Cells as nodes in a graph and edges between cells if they express similar genes or proteins, or are spatially close (more on that later).
  - Social networks : two users are connected if they are friends on a social network

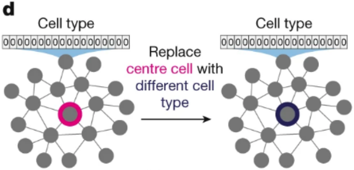


Figure: Nodes here are cells, connected by spatial proximity (from Sun *et al.* Nature. 2025.)

Key notation for talking about graphs

$$A_{ij} = \begin{cases} 1 & \text{there is an edge between nodes } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

Sometimes the individual entries of A_{ij} are replaced with edge weights, reflecting the strength of connection between nods i and j.

Graph embedding problem

- In graph embeddings, we want to find a vector representation for each node that will help us with downstream tasks, like labeling nodes or predicting edges between nodes (link prediction).

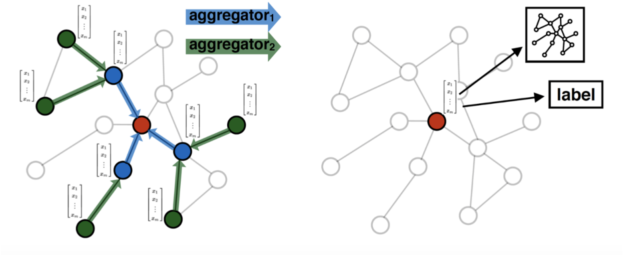


Figure: from `https://snap.stanford.edu/graphsage/`

Graph neural network overview

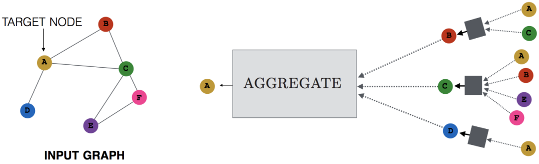


Figure: from `https://www.cs.mcgill.ca/~wlh/grl_book/files/GRL_Book-Chapter_5-GNNs.pdf`. Messages are aggregated from the neighborhood of some target node.

Neural Message Passing

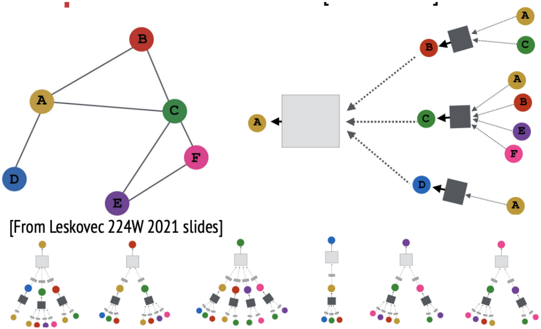
- **Input:** $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a set of node features, $\mathbf{X} \in \mathbb{R}^{d \times |\mathcal{V}|}$
- **Output:** Node embeddings, $\mathbf{z}_u = \mathbf{h}_u^{(K)}, \forall u \in \mathcal{V}$, after K iterations
- **Each Message-Passing Iteration:** A hidden embedding $\mathbf{h}_u^{(k)}$ is updated according to information aggregated from node u's neighborhood, $\mathcal{N}(u)$.

Update Rule

$$\begin{aligned}\mathbf{h}_u^{(k+1)} &= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^{(k)}\left(\left\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\right\}\right)\right) \\ &= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right)\end{aligned}$$

- $\mathbf{m}_{\mathcal{N}_{(u)}}^{(k)}$ is the *message* that is aggregated from u's graph neighborhood, $\mathcal{N}(u)$
- At each iteration, the AGGREGATE function takes as input the set of embeddings of the nodes in u's graph neighborhood, $\mathcal{N}(u)$ and applies it to a previous embedding $\mathbf{h}_u^{(k-1)}$ to generate the updated embedding $\mathbf{h}_u^{(k)}$

Illustrated..



[From Leskovec 224W 2021 slides]

Figure: from https://www.cs.cornell.edu/~arb/slides/2021-03-12-northeastern.pdf

Recent Example Using GNNs to Study Spatial Context

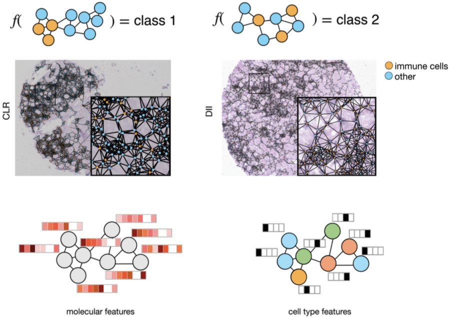


Figure: from https://www.biorxiv.org/content/10.1101/2022.12.08.519537v1.full.pdf. Two colorectal tumor cases (Crohn's-like reaction (CLR) and diffuse inflammatory infiltration (DII)) require spatial patterns to classify correctly.

Ultimate Task - Sample-Level Encodings

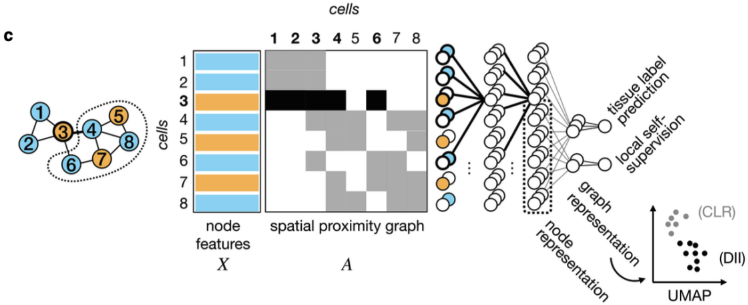


Figure: from Fischer *et al.* 2023. The learned encodings for cell's within a sample can ultimately be averaged to create a pooled feature vector that can separate tumor types.

Step 1 : Spatial Proximity Graph

Define an adjacency matrix, **A** such that with $a_{ij} = 1$ if,

$$||z_i - z_j||_2 < r$$

- z_i is the 2-D location for pixel i
- r is some user-defined radius

Results- Breast Cancer vs Colorectal Cancer

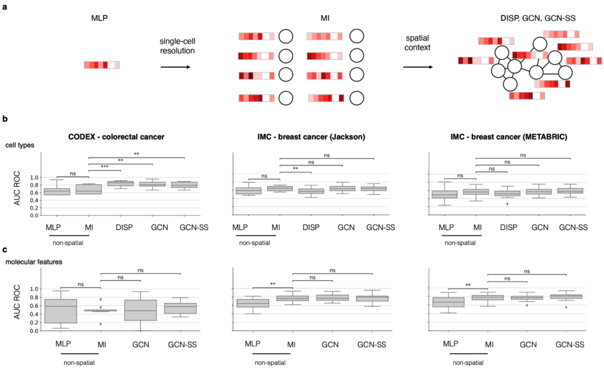
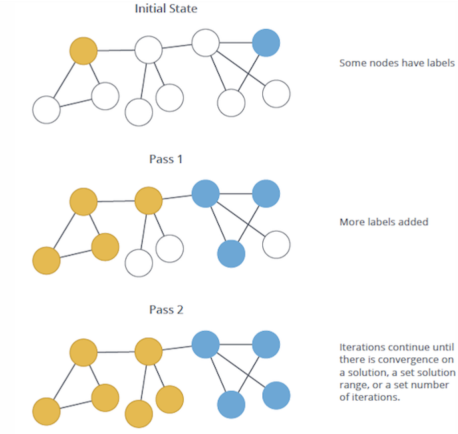


Figure: from Fischer *et al.* 2023. Results compared to just using original features for prediction. Spatial context helps things in the colorectal cancer dataset, but not so much in the breast cancer dataset.

How does this compare to simpler label propagation?

Some of your nodes are labeled, others are unlabeled, and you predict labels of unlabeled nodes based on the structure of the graph.



Initial State

Some nodes have labels

Pass 1

More labels added

Pass 2

Iterations continue until there is convergence on a solution, a set solution range, or a set number of iterations.

Label Propagation Algorithm

Label Propagation Formulation

Consider l labeled and u unlabeled nodes, where each node belongs to one of C classes. First define an $(l+u) \times (l+u)$ probabilistic transition matrix, T as,

$$T_{ij} = P(j \to i) = \frac{w_{ij}}{\sum_{k=1}^{l+u} w_{kj}}$$

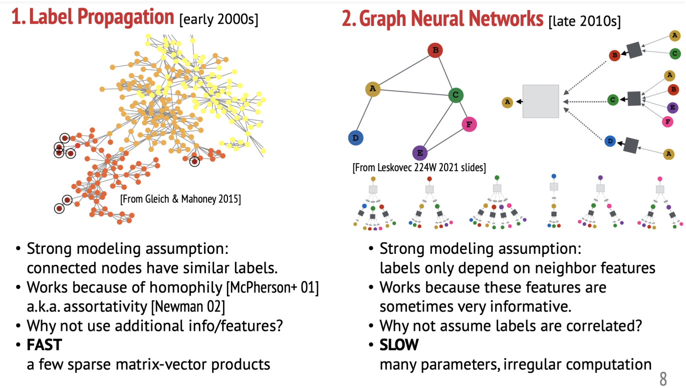
.

Here, W is encoding our weighted adjacency matrix.

Also define a $(l+u) \times C$ label matrix, Y, where the ith row gives the probability toward each of the C clusters.

- $Y \leftarrow TY$. Update until convergence
- Row-normalize Y.
- Clamp the labeled data, or put all of the probability mass on the correct cluster index.

The Debate



1. Label Propagation [early 2000s]

[From Gleich & Mahoney 2015]

- Strong modeling assumption: connected nodes have similar labels.
- Works because of homophily [McPherson+ 01] a.k.a. assortativity [Newman 02]
- Why not use additional info/features?
- **FAST** a few sparse matrix-vector products

2. Graph Neural Networks [late 2010s]

[From Leskovec 224W 2021 slides]

- Strong modeling assumption: labels only depend on neighbor features
- Works because these features are sometimes very informative.
- Why not assume labels are correlated?
- **SLOW** many parameters, irregular computation

8

Figure: from https://www.cs.cornell.edu/~arb/slides/2021-03-12-northeastern.pdf. Tumor labels cannot be inferred based on cell-type frequencies, but instead should glean insights from the overall tissue architecture.

Tradeoffs

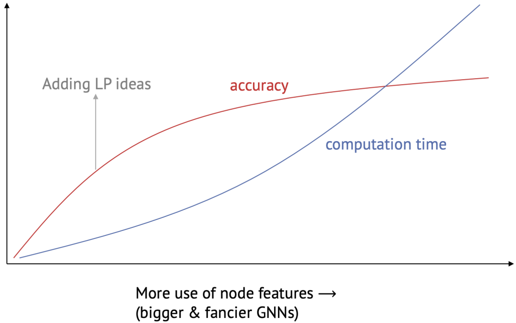


Figure: from `https://www.cs.cornell.edu/~arb/slides/2021-03-12-northeastern.pdf`

Correct and Smooth Approach

- The goal is to compare how a couple of simple methods/intuition can be strung together can be used to classify nodes
- The main idea is to start with a cheap base prediction based on node features (e.g. attributes or coordinates of a spectral embedding), and clean up graph structure through label propagation (**correct and smooth**).

Three Step Process

1. A base prediction made with node features that ignores the graph structure (e.g. with a linear model)
2. A correction step which propagated uncertainties from the training data across the graph to correct the base prediction
3. A smoothing of the predictions over the graph.

Overview of Correct and Smooth Approach

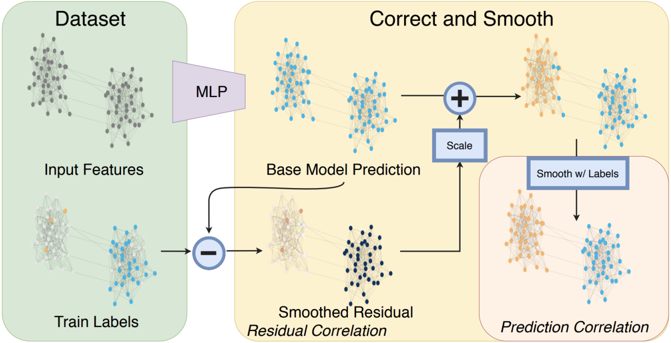


Figure: from Huang *et al.* ICLR. 2021

Tradeoffs

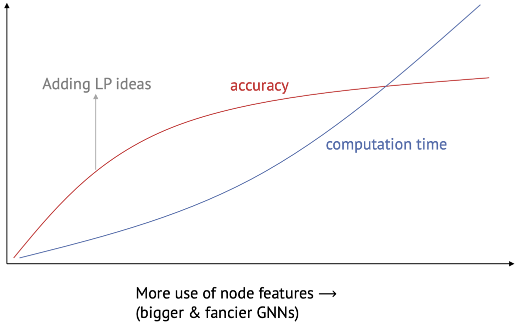


Figure: from `https://www.cs.cornell.edu/~arb/slides/2021-03-12-northeastern.pdf`

Correct and Smooth Approach

- The goal is to compare how a couple of simple methods/intuition can be strung together can be used to classify nodes
- The main idea is to start with a cheap base prediction based on node features (e.g. attributes or coordinates of a spectral embedding), and clean up graph structure through label propagation (**correct and smooth**).

Three Step Process

1. A base prediction made with node features that ignores the graph structure (e.g. with a linear model)
2. A correction step which propagates uncertainties from the training data across the graph to correct the base prediction
3. A smoothing of the predictions over the graph.

Overview of Correct and Smooth Approach

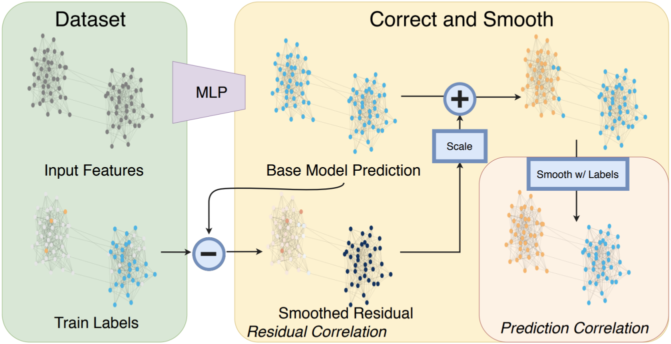


Figure: from Huang *et al.* ICLR. 2021

Notation Preliminaries

- Let there be n nodes.
- Assume we have a feature vector for each node, such that node features are encoded in an $n \times p$ matrix, X.
- Similarly, let A be the adjacency matrix of the graph
- Split nodes into labeled (L) and unlabeled (U) sets
- Define an $n \times c$ matrix, Y with a binary indicator for whether node i is in class c.

Simple Base Predictor

Given the matrix of **features** for each node, X and labels, Y, train a simple model to minimize,

$$\sum_{i \in L_t} \ell\left(f\left(x_i\right), y_i\right)$$

- ℓ is some loss
- Here L_t denotes the set of labeled training nodes
- Specify a matrix, Z containing these base predictions.

Error Correlation - Label Spreading Technique

- The intuition is that errors are expected to be correlated across edges in the graph. Hence, spread uncertainty across the edges.

Define and error matrix, $E \in \mathbb{R}^{n \times c}$ as,

$$E_{L_t,:} = Y_{L_t,:} - Z_{L_t,:}, \quad E_{L_v,:} = 0, \quad E_{U,:} = 0$$

This means that the only non-zero entries are those that correspond to labeled training nodes! These entries represent **residuals**.

Smooth the Error Using a Label Spreading Technique

The errors are smoothed as follow with a label spreading technique,

$$\hat{E} = \underset{W \in \mathbb{R}^{n \times c}}{\arg\min}\ \text{trace}\left(W^T(I - S)W\right) + \mu\|W - E\|_F^2$$

- S is the normalized adjacency matrix, $D^{-1/2}AD^{-1/2}$
- The first term encourages smoothness of the error over the graph
- The second term keeps W close to the initial estimate of error, E.

Our Friend Smoothness and Quadratic Form

We keep seeing the quadratic form come up if we are talking about smoothness. Reminder that,

$$\text{trace}(W^T(I - S)W) = \sum_j w_j^T(I - S)w_j$$

.

- $W \in \mathbb{R}^{n \times c}$
- The first term works out to be, $\sum_{k=1}^{c} \sum_{(i,j) \in E} (W_{ik}/\sqrt{D_{ii}} - W_{jk}/\sqrt{D_{jj}})^2$

Solution

Given

$$\hat{E} = \underset{W \in \mathbb{R}^{n \times c}}{\arg\min} \operatorname{trace}\left(W^T(I - S)W\right) + \mu \|W - E\|_F^2$$

it was previously shown that the solution can be obtained through the following iteration,

$$E^{(t+1)} = (1 - \alpha)E + \alpha S E^{(t)}$$

The quickly converges to $\hat{E}$ and therefore gives corrected predictions as,

$$Z^r = Z + \hat{E}$$

Smoothing Final Predictions with Prediction Correlation

- The next assumption to be used for correction is that adjacent nodes in the graph are likely to have similar labels (e.g. homophily)
- Another round of label propagation will be used to encourage smoothness over distribution of labels.

Starting with the best guess of the labels, H, with $H_{L_t,:} = Y_{L_t,:}$ and $H_{L_v \cup U,:} = Z^{(r)}_{L_v \cup U,:}$, propagate labels as,

$$H^{(t+1)} = (1 - \alpha)H + \alpha S H^{(t)}$$

Final Prediction

The following has now been applied

- Base prediction
- Residual correction
- Label smoothing

After convergence of $H^{(t+1)} = (1-\alpha)H + \alpha S H^{(t)}$, get a final prediction, $\hat{Y} \in \mathbb{R}^{n \times c}$, and assign node to the class with the max predicted probability.

Accuracy vs Number of Parameters

Higher accuracy with less parameters on one of the datasets (and training is also significantly faster)

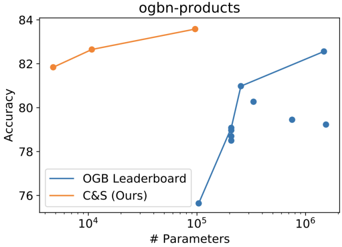


Figure: Result comparing GNN (blue) with correct and smooth approach (orange)