Artificial Intelligence in Bioinformatics

Graph Neural Network Lab report

Di Biase Fabio

.

Investigated problem

The given dataset represents the gene expression profile of 100 patients. The goal of this experiment is to construct a graph-based representation in which each node corresponds to a patient and the edges represent similarities in their gene expression profiles. This graph structure allows us to exploit a Graph Neural Network to leverage the connections between patients with similar gene expression patterns, allowing for accurate node classification. The purpose is to predict the subtype of breast cancer (Luminal A or Luminal B) for each patient by using both their individual gene expression data and their relationships with other patients on the graph.

Building the Graph

The starting point for this analysis is a .csv file containing 100 patients and 1023 features per row. Initially, the dataset is processed to separate the features from the labels, ensuring that the label is properly converted to numerical values. Subsequently, the gene expression values are normalized within the range [0, 1] using a Min-Max Scaler. This normalization step guarantees that each feature contributes proportionally to the subsequent similarity computations, regardless of its original scale.
After preprocessing, the dataset can be represented as a 100×1022 matrix, where each row corresponds to a patient (node) and each column represents a gene expression feature. This matrix serves as the foundation for defining the graph nodes, with each node characterized by its respective gene expression profile. In order to build the edges the Pearson correlation coefficient is used to build a correlation matrix of dimension 100x100:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

This formula represents the pairwise correlation between a node (patient) and another node generating a component of the adjacency matrix representing the graph connectivity (each weight belongs to a [0,1] interval). Since this method produces a fully connected graph, an empirical threshold of 0.2 is applied to cut off weak edges while ensuring that no node remains isolated. Then the main diagonal is brought to 0, removing the self-edges, making visualization clearer.

Graph Convolutional Network

The problem at hand is a node classification task, and the method chosen to address it is the Graph Convolutional Network (GCN). Although more advanced architectures, such as GraphSAGE or Graph Attention Networks (GAT), offer enhanced performance in larger,

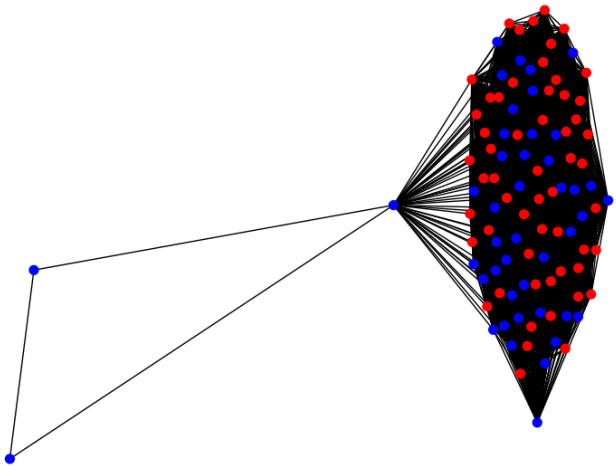

Figure 1: Graph representation of patients based on gene expression similarity. Blue nodes are Luminal A patients while red nodes represent Luminal B patients

dynamical, and more complex datasets, their application here is not necessary given the statical and small set described by the graph. Given the small size of the dataset, the primary focus is on minimizing the number of parameters to prevent overfitting and encourage generalization. The GCN offers a balance between expressiveness and computational efficiency, making it an optimal choice to effectively capture structural relationships within the graph without introducing unnecessary model complexity. More specifically, the chosen architecture has two hidden layers of dimension equal to ten, this configuration was chosen to expand the receptive field and introduce non-linearities while keeping the parameter count manageable. In this context, the GCN is used to propagate and aggregate node feature information throughout the graph, enabling the model to learn meaningful node representations and predict classification labels. The following is the mathematical description of a GCN layer, where $x_w^{(\ell)}$ represents the vector of characteristics of the l-th hidden layer of a node w belonging to the neighborhood of the node v, v included.

$$x_v^{(\ell+1)} = W^{(\ell+1)} \sum_{w \in N(v) \cup \{v\}} \frac{1}{\sqrt{deg(v) \cdot deg(w)}} \cdot x_w^{(\ell)}$$

Considering the graph structure, two or at most three convolutional layers are needed to perform a connection-based classification, and adding more layers will probably lead to overfitting.

GCN Results

The experiments were carried out using a five-fold cross-validation strategy, ensuring that there is no overlap among the test masks. Given the critical nature of the task, evaluating both the F1 score and the accuracy provides a comprehensive assessment of the model's performance. These metrics offer complementary insights into the balance between precision and recall, as well as the overall correctness of predictions. The initial model architecture had two convolutions, showing similar results to the MLP with the exception of a higher variance. Adding another convolutional layer while keeping the same

hyperparameters improves accuracy and reduces variance. The results given in Table 1, represent the best scores achieved during the evaluation.

MLP Comparison

In this section, we compare the GCN results with a simpler and more straightforward approach: classifying each patient using a Multilayer Perceptron (MLP). Unlike the GCN, the MLP does not consider the relationships among patients encoded in the graph structure. Instead, it relies solely on the gene expression values of each patient to predict their label. The network architecture has two hidden layers of dimension equal to ten. The model has no dropout during training since it did not improve validation scores, and 30 epochs were enough to reach the best scores relying only on early stopping for regularization. This comparison allows us to evaluate the impact of incorporating graph connectivity into the prediction process. The results are presented in Table 3 and Table 4.

Fold	Accuracy	F1-Score
Fold 1	85%	0.85
Fold 2	90%	90
Fold 3	80%	0.77
Fold 4	95%	0.95
Fold 5	100%	1.00
Mean	**90%**	**0.89**

Table 1: Best GCN results (torch manual seed 7; k-fold random state 7, dropout 0.3, 100 epochs)

Seed	Mean Accuracy	F1-Score
7	90%	0.89
42	86%	0.85
123	87%	0.85
99	87%	0.85
Mean	**88%**	**0.86**

Table 2: GCN Results on different seeds, both for cross validation and weight initialization

Fold	Accuracy	F1-Score
Fold 1	75%	0.75
Fold 2	95%	0.95
Fold 3	80%	0.81
Fold 4	75%	0.77
Fold 5	100%	1.00
Mean	**85%**	**0.86**

Table 3: Best MLP results (torch manual seed 42 ; k fold random state 42, no dropout, 30 epochs)

Seed	Mean Accuracy	F1-Score
7	78%	0.78
42	85%	0.77
123	85%	0.86
99	83%	0.84
Mean	**83%**	**0.81**

Table 4: MLP Results on different seeds, both for cross validation and weight initialization

conclusion

From these experiments, several observations can be made. The Multilayer Perceptron (MLP) demonstrates relatively stable performances, with accuracy consistently ranging between 78% and 85% in different initialization conditions (e.g., weight initialization and training mask distribution). This stability makes the MLP a reliable baseline for comparison. What we see is the Graph Convolutional Network outperforms the MLP with the same amount of parameters. To reduce the initial GCN model instability, various dropout probabilities were considered and a dropout probability 30% was chosen, different dimensions of the hidden layer were also tried, resulting in an average precision of 88% on different random seeds and obtaining an accuracy of 90% in the best configuration. This suggests that the GCN has significant potential in these types of problems and can deliver high-quality results. Finally, different methods can be tried to further reduce variance in predictions such as ensamble methods and other regularization strategies.

references

https://scikit-learn.org/stable/
https://github.com/senadkurtisi/pytorch-GCN
https://pytorch.org/docs/stable/nn.html
https://pytorch-geometric.readthedocs.io/en/latest/