Report on Graph Neural Network

BARCAROLI Clément

January 2026

1 Definition of a graph

A graph G is the mathematical representation of a network. This network is composed of vertices (or nodes) connected by edges, representing the relationship between the nodes. To begin with, let us introduce V a set of vertices v. V must be finite and non-empty. Each element $\{v \in V\}$ must be distinguishable from the other elements of V but can be of any nature. The second useful set is E, the set of edges. E is defined as a subset of $\{(x, y) \mid (x, y) \in V^2\}$. Each edge is oriented and goes from a point x to a point y. In our case, x is called the tail and y the head of the edge. With this definition, it is well possible that $x = y$: in this case, the corresponding edge is called a loop. Finally, one can define a graph G as $G = (V, E)$. According to Zhou et al (2021)[1], graphs can be categorized according following these criterion :

- Directed or undirected, i.e. if each edge of the graph has a unique direction (like a diode) or not (like a wire).
- Homogeneous or heterogeneous, i.e. if each node is of the same nature (numerical, categorical, discrete, etc) or not.
- Static or dynamic, i.e. if the structure of the graph is constant or not over time.

Any graph can have any combination of these characteristics. From a technical point of view, graphs are represented using two different objects : a graph signal vector denoted V whose elements v_i are the values contained in the edges and an adjacency matrix A describing the link between each edges. The element $\theta_{ij} \in R$ describes the intensity of the connection (or edge) from the i-th edge towards the j-th edge. $\theta_{ij} = 0$ means that there exists no connection from v_i to v_j.

[1] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. *Graph Neural Networks: A Review of Methods and Applications.* arXiv:1812.08434, 2021.

2 Definition of a graph neural network

2.1 Usages

A GNN is a neural network model whose inputs are graphs. It is a generalization of the Convolutional Neural Network architecture, able to perform predictions on data with looser structures than images. There are three types of problem that a GNN model can solve. Node-level tasks aim at predicting the label of a node given the structure of the graph and how the nodes are linked. For example, if we consider a graph representing a road network, with each node being an intersection and the vertices the roads linking them, one could want to predict the amount of car present at a given moment at a given node. Edge level tasks aim at either :

- classifying vertices, for example in terms of importance or recurrence.
- predicting their existence between two nodes. This is utilized by Facebook to do the "People you might know" suggestions.

Lastly, graph-level models aim at predicting general properties on a graph. One could regress the value of a label, classify or match different graphs.

2.2 Construction of GNNs

We now tackle the way such models can be created. As other neural networks, GNN need different types of modules to operate at each layer, given here in their order of operation. The sampling module creates batches to avoid heavy computations which would slow down significantly the evaluation of the model. As it is a common component among machine learning algorithm, it is not treated here. The propagation module is the one transmitting the information gathered in a node to the ones of the next layer. In modern designs, this module is composed of two operators : the convolution operator replacing the older recurrent one, and the skip connection operator. This last operator works exactly like its counterpart in CNN models and thus will not be treated here. At the end of the propagation module, an activation function is used to break any linearity. The pooling module is the last module needed. Its goal is to synthesize the information gathered by the propagation module, which is a non-trivial task when working with graph. As such, its conception is explained here after. To put it simply, each layer of the GNN begins by the sampling module, then the propagation and finally the pooling module. After the last pooling is done, a fully connected layer similar to the ones of MLPs is used to output the final prediction.

2.2.1 Convolution operator

The role of such operators is to filter the data to extract the information available. The convolution operator is no novelty to someone familiar with Convolutional Neural Network. But for CNN, the data are typically well structured

images, whereas here, we expect the operator to work on loosely structured graphs. We need to explain how this generalization works. There are two main approaches to this problem of extraction : either the spectral or the spatial one.

Spectral convolution : As its name suggests, this approach uses spectral decomposition of the graphs. Let us denote x the graph signal i.e. the vector containing the values contained in the nodes of the graph.x belongs to the node space and is as such a vector of real components. The convolution between x and g is denoted $g \star x = \mathscr{F}^{-1}(\mathscr{F}(g) \odot \mathscr{F}(x))$ where $\mathscr{F}$ denotes the Fourrier transform, $\odot$ the element-wise multiplication and g the filter that we want to apply on our data. The Fourrier transform is defined as $\mathscr{F}(x) = U^t x$ where U is the matrix containing the eigenvalues of the Laplace transform of the graph on its diagonals and 0 everywhere else. $\mathscr{F}^{-1}$ is the inverse Fourrier transform and is defined as $\mathscr{F}^{-1}(x) = Ux$. As such, our convolution operation can be rewritten $U(U^t g \odot U^t x))$. This expression means that first, we project x and g on a regular vector space constructed around the graph, before operating a term-wise multiplication and finally turning back to the node space. We use the Fourrier transform as an encryption-decryption operation to allow for standard convolution. Multiple variants exists, each utilizing different gs in order to simplify the convolution operation and lower the computing time.

Spatial convolution The idea behind this approach is to define convolutions for the input graphs specifically instead of projecting them in usual spaces. The choice of such definition is a challenge in itself. The most simple proposed in Zhou et al (2021).[2] comes from Duvenaud et al. (2015)[3]. They use a brute force method, defining a different weight matrix $W^{t+1}_{|\mathcal{N}|}$ for all the degrees $|\mathcal{N}|$ of nodes met during training at layer $t+1$. The degree of a node is the number of nodes in its vicinity. To compute the convoluted values at the step $t+1$, one must first compute $\tilde{h}^t_v = h^t_v + \sum_{u \in N_v} h^t_u$ where h^t_v represents the data contained in the node v at step t, N_v the set of nodes connected to the v node. The term $\tilde{h}^t_v$ is thus a compilation of the information available at step t in the vicinity of the node v. Now, one can compute the updated value taken by this node at step $t+1$, $h^{t+1}_v = \sigma(\tilde{h}^t_v W^{t+1}_{|\mathcal{N}|})$ with σ representing an unspecified activation function. This last step resembles what a MLP would do during the forward phase, the difference being that W is not unique for a given t.

2.2.2 Pooling module

The pooling module of GNNs is also a generalization of the one present in CNN architectures. The pooling module captures general features by compiling the information extracted by the convolution module. The problem with graphs is

[2] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. *Graph Neural Networks: A Review of Methods and Applications.* arXiv:1812.08434, 2021.

[3] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. pp. 2224–2232, 2015.

that their topology does not allow for a local pooling, because local structures are irregular. There exist two ways of tackling this problem.

Direct pooling : The information extracted by the convolutions is aggregated for each node, using maximum, average and more complex methods. Formally, after having performed T message passing, we get the set $H^T = \{h_v^T \, v \in V\}$ containing all the data transformed by T convolutions. We introduce $\mathscr{R} : \{h_v^T\}_{v \in V} \to h_G$ our pooling function. Here h_G represents the pooled information for a given node. For $\mathscr{R}$ to be a valid candidate, it must be invariant to permutations on H^T i.e. the order of the h_v^T must play no role. For example, one could define $\mathscr{R}$ as an average. This approach does not reduce the size of the data and treats each node individually, ignoring the relationship between the nodes. This simplistic method is the simplest to use and in cases where dimension is already small, not reducing it poses no problem.

Hierarchical pooling : This approach tries to mitigate the weaknesses of direct pooling by aggregating the information between each message passing. Formally, at step t, we have $G_t = (V_t, A_t)$ and H_t its representation after message passing. The goal of hierarchical pooling is to define a way of transforming G_t into G_{t+1} and H_t into H_{t+1} under the constraint $|V_{t+1}| < |V_t|$. This constraint means that we reduce at each step the number of vertices in the representation of our graph. To do so, we must introduce an assignation function $S_t \in R^{|V_t| \times |V_{t+1}|}$ and then $H_{t+1} = S_t^T H_t$ and $A_{t+1} = S_t^T A_t S_t$. This definition of pooling is more complex than what is usually done for CNN (max, average,...) but achieves the same goal of reducing the size of the computed objects while synthesizing information. For GNN, the size reduction is crucial as computations time is significant and increases with the number of vertices.

3 Conclusion

Graph neural networks are an extension of Convolution neural networks, capable of extracting information from loose structures of data. They are useful in numerous domains but especially in molecular chemistry : from the molecular structure of a molecule, one can predict crucial properties, even before synthesizing said molecule. To do so, such models use convolution modules, using edges to define proximity between vertices that contain the features. The pooling phase helps concentrating the gathered information before passing it to the next layer. The final result is outputted by a MLP.