The Graph Neural Network Model

November 26, 2023

Abstract

A remarkable number of ideas across various domains can be effectively depicted as graphs, ranging from social networks and railway maps to molecules. Graphs offer an elegant way to abstract these concepts, making them a primary tool for representing data. This abstraction not only highlights the characteristics of individual data points but also captures the relationships between them. However, this versatility poses a challenge in the field of machine learning, as desining algorithms that efficiently handle such interconnected data has proven to be a challenging task.

1 Introduction

We shall start by giving a brief introduction to out mathematical model. A graph $\boldsymbol{G}$ is a pair $(\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V}$ is the set of vertices, and $\boldsymbol{E} \subseteq$

$\boldsymbol{V} \times \boldsymbol{V}$ is the set of edges, representing the relationships between the vertices. An example of a graph would be a railway map, where the vertices are the stations and the edges are the railways connecting them. Often we're interested in giving some attributes to the edges. In the case of the railway map, we could assign the length of the railway to the edges, in which case the graph would be a weighted graph. On the other hand, edges could be multidemensional, in which case the graph would be a hypergraph, which is especially useful for representing molecules, where the vertices are the atoms and the edges are the bonds between them.

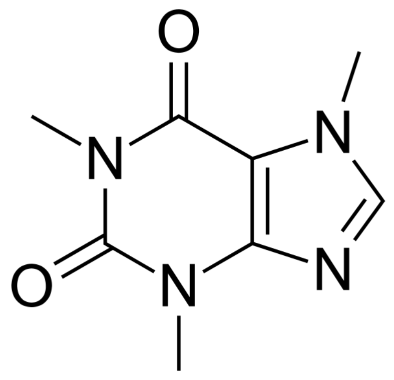

Figure 1: A molecule of caffeine, represented as a graph, where the vertices are the atoms and the edges are the bonds between them.

While providing the definitions, we will stick to unweighted and undirected, i.e. simple, graphs. The model can be easily extended to other cases. For a vertex $\boldsymbol{v} \in \boldsymbol{V}$, the set $\boldsymbol{N(v)} = \{\boldsymbol{u} \in \boldsymbol{V} | (\boldsymbol{u}, \boldsymbol{v}) \in \boldsymbol{E}\}$ is called the neighborhood of $\boldsymbol{v}$, while $\boldsymbol{E(v)} = \{(\boldsymbol{u}, \boldsymbol{v}) \in \boldsymbol{E}\}$. The degree of a vertex $\boldsymbol{v}$ is $|\boldsymbol{N(v)}|$, i.e. the number of vertices adjacent to $\boldsymbol{v}$.

For any meaningful application of graphs in data science and machine learning, we shall assign labels (real vectors) to the vertices and the edges. They will be represented by $\boldsymbol{l_v} \in \mathrm{R}^{\boldsymbol{l_V}}$ and $\boldsymbol{l_{(v,u)}} \in \mathrm{R}^{\boldsymbol{l_E}}$, respectively.

2 The Graph Neural Network Model

In this section, we will present the Graph Neural Network model, which is a powerful algorithm for learning on graphs, first introduced by Scarselli et al. [1].

The novelty of the model lies in the fact that it is able to learn the relationships between the vertices of a graph, as the representation of each vertex depends on the representations of its neighbors.

More formally, assume we are given a dataset of graphs:
$\boldsymbol{\mathcal{D}} = \{(\boldsymbol{G_i}, \boldsymbol{v_{i,j}}, \boldsymbol{t_{i,j}}) | \boldsymbol{G_i} = (\boldsymbol{V_i}, \boldsymbol{E_i}), \boldsymbol{v_{i,j}} \in \boldsymbol{V_i}, \boldsymbol{t_{i,j}} \in \mathrm{R}^{\boldsymbol{m}}\}$,
where $\boldsymbol{v_{i,j}}$ is the $\boldsymbol{j}$-th vertex of the $\boldsymbol{i}$-th graph, and $\boldsymbol{t_{i,j}}$ is the target label of $\boldsymbol{v_{i,j}}$. The objective is to learn a function $\boldsymbol{\varphi} : \boldsymbol{G} \times \boldsymbol{V} \to \mathrm{R}^{\boldsymbol{m}}$ such that $\boldsymbol{\varphi}(\boldsymbol{G_i}, \boldsymbol{v_{i,j}}) \approx \boldsymbol{t_{i,j}}$, where $\boldsymbol{G}$ and $\boldsymbol{V}$ are the set of all graphs and vertices, respectively, considered in $\boldsymbol{\mathcal{D}}$.

The underlying intuitive idea of this algorithm is that in a graph, vertices are representing object, data points, etc., while the edges are representing the relationships between them. Therefore, the representation of a vertex should depend on the representations of its neighbors. To mimic this, we will attach a ***state*** to each vertex, $\boldsymbol{x_v} \in \mathrm{R}^{\boldsymbol{s}}$, which is encodes information contained in $\boldsymbol{N(v)}$, and is used to compute an ***output*** $\boldsymbol{o_v} \in \mathrm{R}^{\boldsymbol{m}}$, s.t. $\boldsymbol{o_v \approx t_v}$.

It is natural to express the dependence of $\boldsymbol{x_v}$ on its neighbors as a function, so let $\boldsymbol{f_w}$ be a parametric function, called ***local transition function***. Similarly, let $\boldsymbol{g_w}$ be the ***local output function***. Then, the state and output of a vertex $\boldsymbol{v}$ are computed as follows:

$$\boldsymbol{x_v = f_w(l_v, l_{N(v)}, l_{E(v)}, x_{N(v)})} \tag{1}$$

$$\boldsymbol{o_v = g_w(x_v, l_v)} \tag{2}$$

, where $\boldsymbol{l_v}$, $\boldsymbol{l_{N(v)}}$, $\boldsymbol{l_{E(v)}}$ are the labels of $\boldsymbol{v}$, $\boldsymbol{N(v)}$, $\boldsymbol{E(v)}$, respectively, stacked together, and $\boldsymbol{x_{N(v)}}$ the states of the neighbors of $\boldsymbol{v}$, stacked.

It is worth noting, nevertheless, that the use of $\boldsymbol{l_{N(v)}}$ and $\boldsymbol{l_{E(v)}}$ may be considered redundant, as $\boldsymbol{x_{N(v)}}$ already encodes this information, while $\boldsymbol{l_v}$ is contained in $\boldsymbol{x_v}$ for $\boldsymbol{g_w}$. This would be just a similar definition to $\boldsymbol{f_w}$ and $\boldsymbol{g_w}$.

Let $\boldsymbol{x}$, $\boldsymbol{o}$, $\boldsymbol{l_V}$, and $\boldsymbol{l_E}$ be the vectors constructed by stacking $\boldsymbol{x_v}$, $\boldsymbol{o_v}$, $\boldsymbol{l_v}$, and $\boldsymbol{l_{(v,u)}}$ respectively, for all $\boldsymbol{v} \in \boldsymbol{V}$ and $\boldsymbol{(v,u)} \in \boldsymbol{E}$. Then, (1) and (2) can be rewritten as:

$$\boldsymbol{x = F_w(x, l_V, l_E)} \tag{3}$$

$$\boldsymbol{o = G_w(x, l_V)} \tag{4}$$

, and we therefore call $\boldsymbol{F_w}$ and $\boldsymbol{G_w}$ the ***global transition function*** and

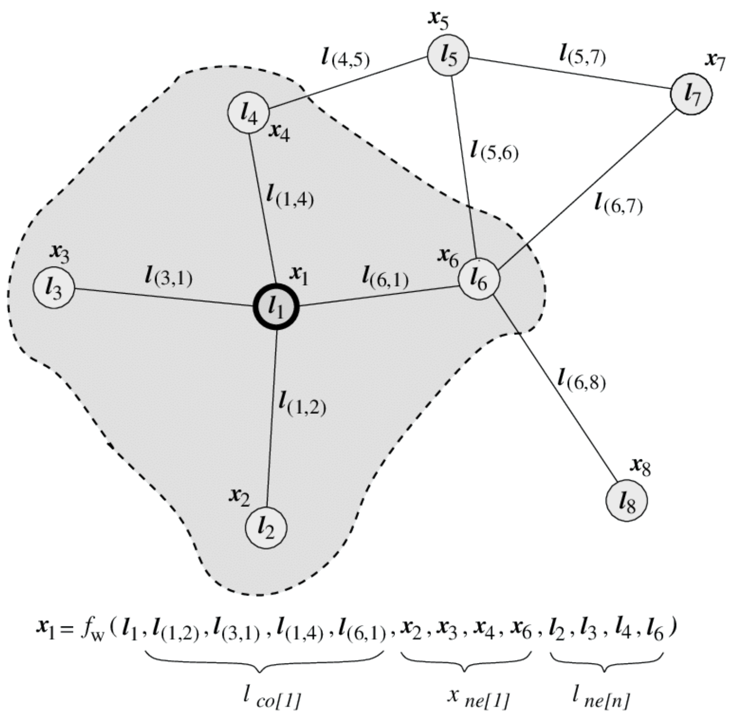

Figure 2: A graph and the neighborhood of a node. The state $\boldsymbol{x_1}$ of node 1 depends on the states of its neighbors, $\boldsymbol{x_2}$, $\boldsymbol{x_3}$, $\boldsymbol{x_4}$, and $\boldsymbol{x_6}$.

global output function, respectively.

3 Results

This is the results section of your article.

4 Conclusion

This is the conclusion section of your article.

References

[1] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. IEEE Transactions on Neural Networks, 20(1), 61-80.