Master Thesis

Federico Calabrò 247976

Department of Mathematics and Computer Science
2024

Contents

1 Neural Networks

1.1 Graph Neural Networks

2 Graph Data Structure

A *graph* G [Figure 1] is a non-linear data structure consisting of vertices and edges. The vertices are sometimes also referred to as nodes and the edges are arcs that connect any two nodes in the graph. Graphs are used to represent relationships between different entities and have applications in many fields including Computer Science, Physics, Biology, Chemistry, Optimization Theory and many more.

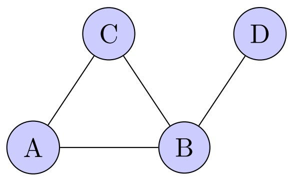


Figure 1: A simple graph

2.1 Formal Definition

A graph is formally defined as a tuple $G = (V, E)$, where:

- V is a finite set of vertices (or nodes).
- E is a set of edges, where each edge is an unordered pair of distinct vertices from V. Thus, $E \subseteq \{\{u, v\} \mid u, v \in V \text{ and } u \neq v\}$.

2.2 Types of Graphs

Graphs can be classified into various types based on their properties, including:

- **Directed Graph** [Figure 2]: A graph in which the edges have a direction, i.e., each edge is an ordered pair of vertices.

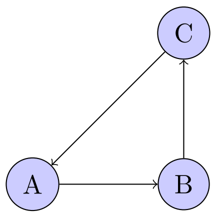


Figure 2: Directed graph example.

- **Undirected Graph** [Figure 3]: A graph in which the edges do not have a direction, i.e., each edge is an unordered pair of vertices.

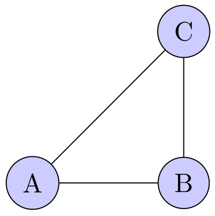


Figure 3: Undirected graph example.

- **Weighted Graph** [Figure 4]: A graph in which a weight (or cost) is associated with each edge.

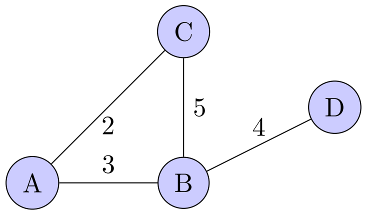


Figure 4: Weighted graph example.

- **Simple Graph** [Figure 5]: A graph with no loops (edges connecting a vertex to itself) and no multiple edges (more than one edge connecting the same pair of vertices).

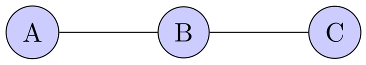


Figure 5: Simple graph example.

- **Complete Graph** [Figure 6]: A graph in which there is exactly one edge between each pair of distinct vertices.

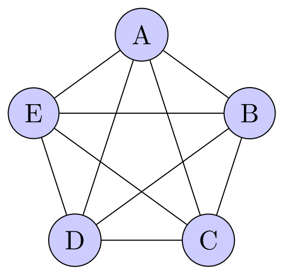


Figure 6: Complete graph example.

- **Bipartite Graph** [Figure 7]: A graph whose vertices can be divided into two disjoint sets U and W such that every edge connects a vertex in U to a vertex in W.

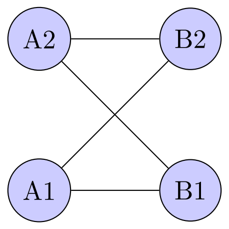


Figure 7: Bipartite graph example.

2.3 Graph Representation

Graphs can be represented in various ways, including:

- **Adjacency Matrix** [Figure 8]: A square matrix A of size $|V| \times |V|$ where $A_{ij} = 1$ if there is an edge between vertices v_i and v_j, and $A_{ij} = 0$ otherwise.

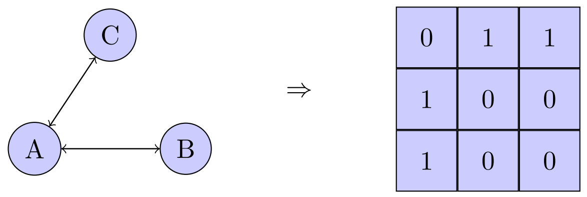
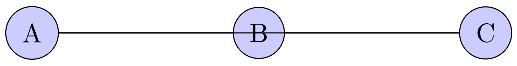


Figure 8: Adjacency Matrix representation.

- **Adjacency List** [Figure 9]: An array of lists. The array contains a list for each vertex, and each list contains the vertices that are adjacent to the corresponding vertex.



$$[\ Adj(A) = [B], \quad Adj(B) = [A, C], \quad Adj(C) = [B]\ ]$$

Figure 9: Adjacency List representation for an undirected graph.

3 Graph Similarity Problem

The *graph similarity problem* involves determining the degree of similarity between two graphs. This problem has numerous applications in pattern recognition, computer vision, bioinformatics, and other fields. One common method to quantify graph similarity is through the *graph edit distance* (GED).

3.1 Graph Edit Distance (GED)

The *graph edit distance* between two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is defined as the minimum cost required to transform G_1 into G_2 using a sequence of atomic operations. Formally, let Σ be the set of all possible edit operations, and let $c : \Sigma \to \mathbb{R}^+$ be a cost function that assigns a positive real number to each operation. The GED is then given by:

$$\mathrm{GED}(G_1, G_2) = \min_{\sigma \in \Sigma^*} \sum_{o \in \sigma} c(o)$$

where Σ^* denotes the set of all finite sequences of operations from Σ, and o represents an individual operation within a sequence σ.

The computation of GED is known to be **NP-HARD** [1], indicating that finding the exact minimum edit distance between two graphs is computationally intensive.

3.2 Atomic Operations

The basic atomic operations typically includes:

- **Vertex Insertion**: Inserting a new vertex v into the graph.
- **Vertex Deletion**: Deleting an existing vertex v from the graph.
- **Edge Insertion**: Inserting a new edge $e = \{u, v\}$ into the graph.
- **Edge Deletion**: Deleting an existing edge $e = \{u, v\}$ from the graph.

To demonstrate the Graph Edit Distance (GED), consider pair of graphs represented in 10:

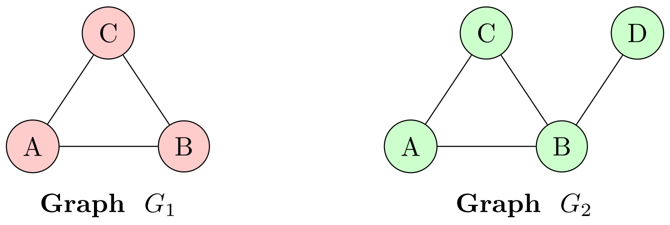


Figure 10: Pair of graphs to demonstre Graph Edit Distance.

In this example, graph G_1 has vertex set $V_1 = \{A, B, C\}$ and edge set $E_1 = \{\{A, B\}, \{A, C\}, \{B, C\}\}$, while graph G_2 has vertex set $V_2 = \{A, B, C, D\}$ and edge set $E_2 = \{\{A, B\}, \{A, C\}, \{B, C\}, \{B, D\}\}$. The transformation (which cost is lowest) from G_1 to G_2 involves: Inserting the vertex D and Inserting the edge $\{B, D\}$. If we assign a cost of 1 to each operation the total cost (GED) is: $1 + 1 = 2$.

4 SimGNN

5 Bottleneck Loading GPU Problem

6 Dataset Generation

7 State of the Art Review

References

[1] Wikipedia. Np-hard, 2024. https://en.wikipedia.org/wiki/NP-hardness.