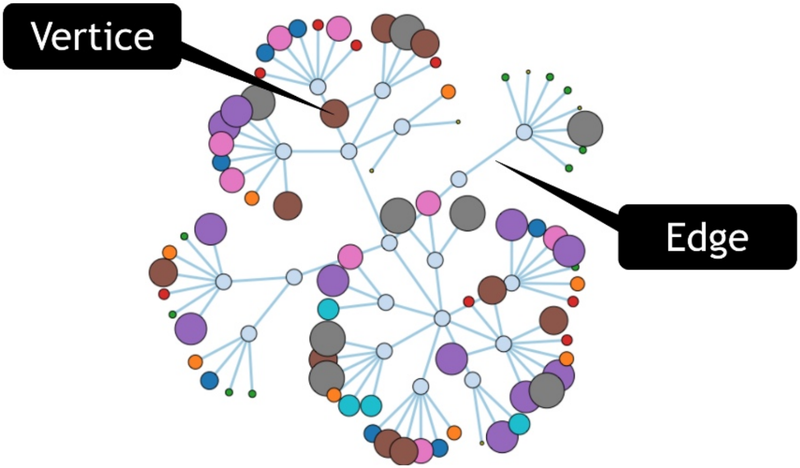
Graph Neural Networks

GNN



Многие типы данных можно представить в виде графа

Image credit: Medium

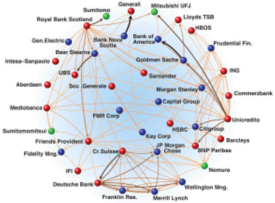


Image credit: Science

Image credit: Lumen Learning

Social Networks **Economic Networks** **Communication Networ**

Image credit: Missoula Current News

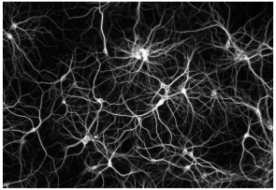


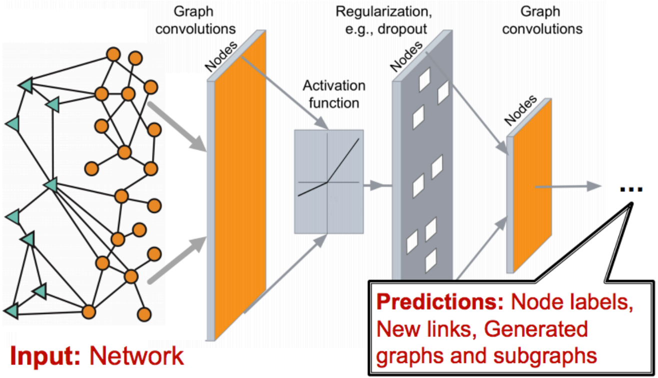
Image credit: The Conversation

Risk
modelling.research

В чем сложность в графовых типах данных

- Большие размеры порождают большое количество связей
- Сложная топологическая структура
- Нет четкого порядка в вершинах графа
- Часто имеют динамическую структуру

Какие задачи можно решать с помощью GNN



Какие задачи можно решать с помощью GNN

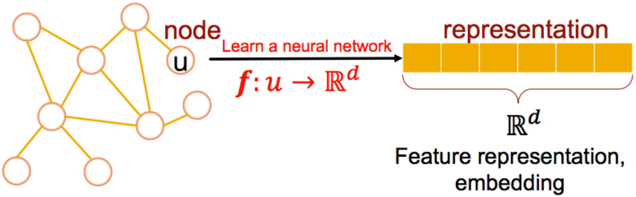
Пусть $G = (V, E)$ граф, где X_v представляет собой вершину графа $v \in V$

1. Задача классификации вершин (Node classification): каждой вершине графа $v \in V$ и ее метки y_v ставится в соответсвие такой вектор h_v, который описывал бы вершину v так, что $y_v = f(h_v)$.
2. Задача классификации графов (Graph classification): для множества графов $G_1, ..., G_N \in G$ и их меток $y_1, ..., y_N$ ставится в соответствие вектор h_G, который способен предсказать метку всему графу $y_G = g(h_G)$.

Risk modelling.research

Основная идея

GNN использует структуру графа и характеристики его вершин, чтобы построить векторное представление для каждой вершины или для всего графа.



Архитектура сети

1. Используя матрицу смежности, определяем соседей для каждой вершины графа.
2. Итеративно учит векторное представление на основе агрегации соседей.

$$a_v^{(k)} = AGGREGATE^{(k)}(\{h_u^{(k-1)} : u \in \mathsf{N}(v)\}),$$

$$h_v^{(k)} = COMBINE^{(k)}(h_v^{(k-1)}, a_v^{(k)})$$

На k-м слое строим векторное представление $h_v^{(k)}$ для вершины v, где $\mathsf{N}(v)$ - это множество смежных вершин.

Risk modelling.research

AGGREGATE

В качестве функции агрегации могут выступать:

1. Max pooling
2. Average pooling
3. RNN (инвариантность к порядку)
4. Convolution

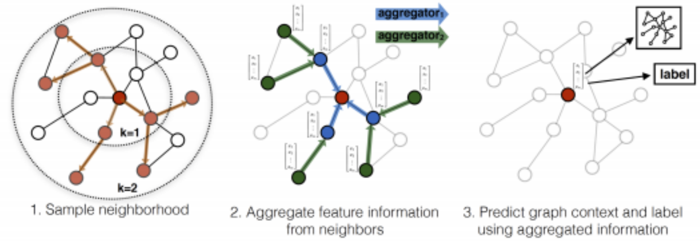


Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

GraphSAGE

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \to 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
    for $v \in \mathcal{V}$ **do**
        $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
        $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
    end
    $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

Risk
modelling.research

Статьи

1. GraphSAGE https://arxiv.org/pdf/1706.02216.pdf
2. Graph Attention Networks https://arxiv.org/pdf/1710.10903.pdf
3. GCN https://arxiv.org/pdf/1802.08888.pdf, https://arxiv.org/pdf/1902.07153.pdf
4. https://arxiv.org/pdf/1810.00826.pdf

Risk
modelling.research