Graphical Neural Network(GNN)

2025年11月7日 星期五 21:06

Background Knowledge:

1. Distributed Vector Transformation:
   - $Local\ (one - hot)\ representation\ (size\ V) \Rightarrow Distributed\ representation\ (size\ D)$
   - $\boxed{r = EI_w}$
     $E = trainable\ embedding\ matrix$
     $I_w = local\ reresentation$
     $r = output\ vector$
   - For example:

$$\alpha_{dog,cat,monkey} = [dog \quad cat \quad monkey] = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \alpha \subset \mathbb{R}^{4\times 3}$$

$$E = \begin{bmatrix} e_{11} & e_{12} & e_{13} & e_{14} \\ e_{21} & e_{22} & e_{23} & e_{24} \\ e_{31} & e_{32} & e_{33} & e_{34} \end{bmatrix} = \begin{bmatrix} weights\ for\ dimension\ 1 \\ weights\ for\ dimension\ 2 \\ weights\ for\ dimension\ 3 \end{bmatrix}, E \subset \mathbb{R}^{3\times 4}$$

$$r = E\alpha = \begin{bmatrix} e_{11} & e_{12} & e_{13} & e_{14} \\ e_{21} & e_{22} & e_{23} & e_{24} \\ e_{31} & e_{32} & e_{33} & e_{34} \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} e_{14} & e_{12} & e_{11} \\ e_{24} & e_{22} & e_{21} \\ e_{34} & e_{32} & e_{31} \end{bmatrix}, r \subset \mathbb{R}^{3\times 3}$$

   - How to find E?
     In general, neural network training, which has different approached (mentioned in different note)
   - What does this mean?
     Transforms discrete symbolic representations into continuous and meaningful distributed representation, which capture the relationships between concepts.
     Compress the dimensions => make processing easier

2. Gradient
   - Definition:
     The slope of the error curve at a specific point, a multidimensional compass that points toward lower error.
   - Loss function:
     $L(w_1, w_2, w_3, \ldots, w - n), where\ w_n\ are\ all\ the\ weights\ in\ the\ network.$

$$\boxed{Gradient = \left[\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial w_3}, \ldots, \frac{\partial L}{\partial w_n}\right]}, contains\ the\ partial\ derivative\ of\ loss\ function$$

   - Gradient Vanishing: when these partial derivatives become extremely small
   - Gradient Descent: using these to "descend" the error mountain

GNN

1. What is GNN?
   - Neural models trying to find the dependencies between input graphs through passing the message between nodes of graphs

2. General Design of GNN

- Denote the graph as
  $G = (V, E)$
  $|V| = N, the\ number\ of\ nodes\ (A, B, \ldots G) in\ the\ graph$
  $|E| = N^e, the\ number\ of\ edges$

- Neural Message Passing
  $current\ neibor\ states \longrightarrow prepare$ message e.g. $\mathrm{A = f(D \Rightarrow F), B = f(E \Rightarrow F)}$
  $\longrightarrow Summarize\ recieved\ information \longrightarrow$
  $Current\ Node\ State\ h^n_{(t-1)} \longrightarrow h^t_n = q\left(h^n_{(t-1)}, x\right) \longrightarrow Next\ Node\ State\ h^t_n$

- General Equation representing the logic of **Message Passing**

$$\boxed{h^t_n = q\left(h^n_{(t-1)}, \cup_{k \forall\, n_j: n \to n_j} f_t(h^n_{(t-1)}, k, h^{n_j}_{(t-1)})\right)}$$

  $h^{n_j}_{(t-1)}: Current\ state\ of\ neighbor\ node\ n_j$
  $\cup$ means the aggregation over all neighbors n_j
  $f_t: Message\ function$
  $q: Update\ function$
  $\cup: Permutaion\ - invariant\ aggregation\ operator$ (sum, max/min, attention-weighted combination)

- GRU Fights Gradient Vanishing:
  Decide what to remember and what to forget when processing sequential data.
  $\boxed{h^t_n = GRU\left(h^n_{(t-1)}, m\right)}$, update the next node's state using GRU mechanism

$$\boxed{m = \sum_{\substack{k \\ \forall n_j: n \to n_j}} E_k\, h^{n_j}_{(t-1)}}, m\ is\ the\ FINAL\ aggregated\ message\ from\ neighborhood$$

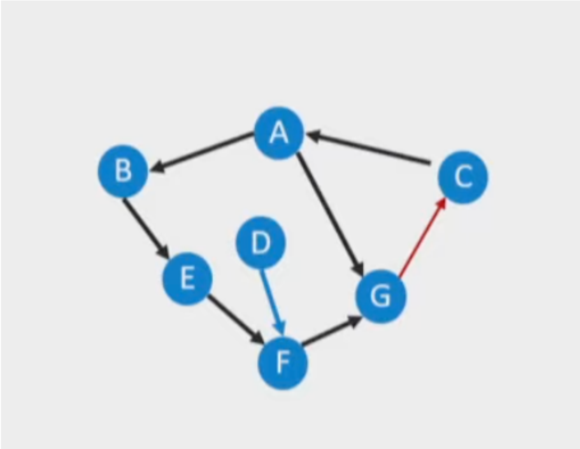
  E_K: a learnable weight matrix, which transforms neighbor information on edge type k
  k: egde type index (friend, family, colleague)

$$\boxed{If \quad h^{n_j}_{(t-1)} \subset \mathbb{R}^{1 \times D}, \qquad E_k \subset \mathbb{R}^{D \times \mathrm{D}}}$$

- Adjacency Matrix (A) -- How message between each node start flowing
  According to the graph:
  In general

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad \mathrm{N} = \begin{bmatrix} \mathrm{a} \\ \mathrm{b} \\ \mathrm{c} \\ \mathrm{d} \\ \mathrm{e} \\ \mathrm{f} \\ \mathrm{g} \end{bmatrix}, \text{representing each node}$$

  $A_{ij} = 1\ when\ there\ is\ an\ edge\ from\ node\ i\ to\ node\ j$
  $A_{ij} = 0\ when\ there's\ no\ egde\ like\ above$



$$\dot{A}N = \begin{bmatrix} b+g \\ e \\ a \\ f \\ f \\ g \\ c \end{bmatrix}, this\ is\ Message\ Aggregation, \boxed{m = \sum_{j \in N(i)} h_j}.$$

AN represents the message received, it will experience either update function or gating (like GRU) afterward.

- GNN in Matrix Operation: number of nodes = V

$$1.\ Node\ States: H_t = \begin{bmatrix} h_t^{n_0} \\ \dots \\ h_t^{n_K} \end{bmatrix}, H_t \subset \mathbb{R}^{V \times D}$$

$$2. Message\ to\ be\ sent: M_t^k = E_k H_t, M_t^k \subset \mathbb{R}^{V \times M}$$

$$3. Received\ Messages: R_t = \sum_k AM_t^k,\ R_t \subset \mathbb{R}^{V \times M}$$

$$4. Update\ H_{t+1} = GRU(H_t, R_t)$$

- Related Python Skills:
- Tensor operations: np.random.rand(rows, columns)
- Result feature: np.einsum()
- Example in feature transformation:
-
```
H = np.random.rand(4,6) #4 nodes, 6 features -- representing inital features of nodes
W = np.random.rand(6,6) #Weight matrix -- representing the relations we want
A = np.random.rand(4,4) #Adjacency matrix -- representing adjacency between each node

#Step 1: H @ W, transformation of node features
#Step 2: A @ H, neiborhood aggregation
#Step 3: Step 1 @ Step 2, representing the features under certain context defined by W

#Implementation
result = np.einsum('ab,bc,ac->ab',H,W,A @ H)

print(result)
Last executed at 2025-11-09 23:52:29 in 7ms
[[0.00582522 0.34805634 0.87707221 0.07009528 1.00984862 0.57468148]
 [1.94699754 0.95089662 0.71370607 1.48573596 0.77851049 1.19245227]
 [2.1654061  2.24591608 1.84443366 0.03855653 0.33354836 1.53125122]
 [1.44694365 2.12301669 2.48225943 1.23538871 1.65580911 1.75077962]]
```