Food Recommender Based on GCN and MLP

Jingshuai Qian

April 27, 2025

1 Load the model

Code to load and apply the recommender model: `GAT_MLP/food_recommender_apply.py`

2 Algorithm demonstration

2.1 Architecture

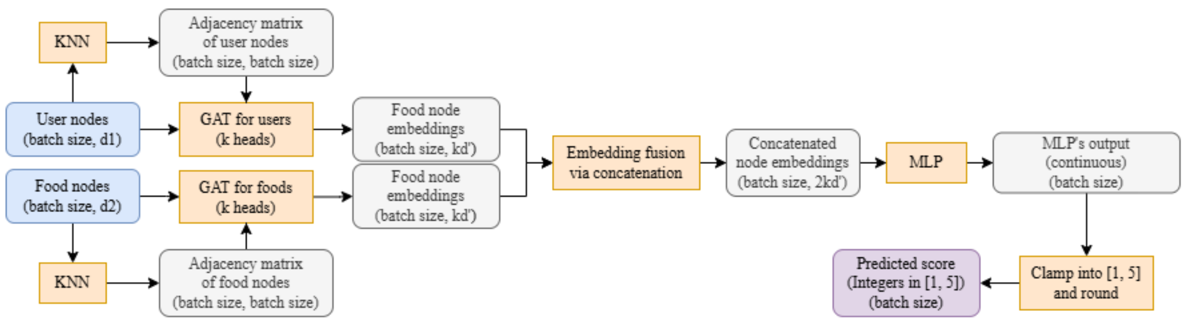

Figure 1: The two stage architecture of food recommender

2.2 Algorithms

2.2.1 KNN

Algorithm 1 Compute adjacency matrix through KNN

Input: $X \in \mathbb{R}^{n \times d}$ (Node feature matrix), k (Number of neighbors)
Output: A^{norm} (Normalized adjacency matrix)

For each pair of nodes, calculate their Euclidean distance $S_{ij} = \sqrt{\sum_{k=1}^{d}(x_{ik} - x_{jk})^2}$, where $i, j \in \{1, \ldots, n\}$ and $i \neq j$.

For each node $i \in \{1, \ldots, n\}$, choose $(k+1)$ nearest neighbors as $\mathcal{N}_i = \{j_1, j_2, \ldots, j_{k+1}\}$ where $S_{ij_1} \geq S_{ij_2} \geq \cdots \geq S_{ij_{k+1}}$.

Initialize the adjacency matrix as $A^{\text{raw}} \in \{0, 1\}^{n \times n}$ where $A^{\text{raw}}_{ij} = 1$ if $j \in \mathcal{N}_i$ and $i \neq j$, otherwise $A^{\text{raw}}_{ij} = 0$.

Symmetrize the adjacency matrix by taking the element-wise maximum, i.e. $A^{\text{sym}}_{ij} = \max(A^{\text{raw}}_{ij}, A^{\text{raw}}_{ji})$. ▷ To ensure the undirected graph property.

Compute the degree matrix D where $D_{ii} = \sum_{j=1}^{n} A^{\text{sym}}_{ij}$ and $D_{ij} = 0$ where $i, j \in \{1, \ldots, n\}$ and $i \neq j$.

Normalize the adjacency matrix: $A^{\text{norm}} = D^{-1/2}(A^{\text{sym}} + I)D^{-1/2}$.
return A^{norm}

2.2.2 GAT

Algorithm 2 Graph Attention Network (GAT) Forward Propagation

Input: $X \in \mathbb{R}^{N\times d}$ (Node feature matrix); $A \in \{0,1\}^{N\times N}$ (Adjacency matrix) K (Number of attention heads); d' (Output dimension per head); α (LeakyReLU negative slope)
Output: $Z \in \mathbb{R}^{N\times Kd'}$ (Node embeddings)

Initialize $Z \leftarrow \text{EmptyMatrix}(N, Kd')$
for $k = 1$ to K **do**
  $W^k \leftarrow \text{LearnableMatrix}(d \times d')$
  $h^k \leftarrow XW^k$ ▷ Linear projection
  $a^k \leftarrow \text{LearnableVector}(2d')$ ▷ Compute attention coefficients
  $E^k \leftarrow \text{ZeroMatrix}(N \times N)$
  for $i = 1$ to N **do**
    for $j = 1$ to N **do**
      if $A_{ij} = 1$ **or** $i = j$ **then**
        $e^k_{ij} \leftarrow \text{LeakyReLU}\big(a^k[h^k_i \| h^k_j]\big)$ ▷ Calculate the neighbor pair's attention coefficient, where "$\|$" means concatenation
        $E^k_{ij} \leftarrow e^k_{ij}$
      else
        $E^k_{ij} \leftarrow -\infty$ ▷ This pair of nodes are not neighbors so they do not have an attention coefficient
      end if
    end for
  end for
  $\alpha^k \leftarrow \text{softmax}(E^k, \text{dim} = 1)$ ▷ Normalize attention weights using row-wise softmax
  $z^k \leftarrow \alpha^k h^k$ ▷ Aggregate neighbors
  $Z[:, (k-1)d' : kd'] \leftarrow z^k$ ▷ Concatenate heads as the final embedding
end for
return Z

2.2.3 MLP

Algorithm 3 Forward propagation through MLP

Input: $Z \in \mathbb{R}^{n\times d_{\text{emb}}}$ (Node embedding matrix), h (Number of hidden layers) $L \in \mathbb{R}^h$ (Hidden layers' dimensions)
Output: $\hat{r}_{\text{valid}} \in \mathbb{R}^n$ (Predicted scores)

$H^{(0)} = Z$.
for $l \in \{1, \dots, h\}$ **do**
  Propagation on the current hidden layer: $H^{(l)} = \text{ReLU}(W^{(l)}H^{(l-1)} + b^{(l)})$, where $W^{(l)} \in \mathbb{R}^{L_l \times L_{(l-1)}}$ and $b^{(l)} \in \mathbb{R}^{L_l}$.
end for
Convolution on the output layer: $\hat{r} = W^{(h+1)}H^{(h)} + b^{(h+1)}$, where $W^{(h+1)} \in \mathbb{R}^{L_h \times n}$ and $b^{(h+1)} \in \mathbb{R}^n$.
Transform all predicted scores to integers in $[1, 5]$: $\hat{r}_{\text{valid}} = \text{round}(\text{clamp}(\hat{r}, 1, 5))$.
return $\hat{r}_{\text{valid}}$

3 Test results of the best model

MSE Loss=1.1760, RMSE=1.1323, MAE=0.8619