Contents

Part I

Graph Convolutional Neural Network (GCN)

Paper

1 Problem

Given dataset $X \in \mathbb{R}^{n \times d}$ where n is the number of nodes, d is the number of features. We want to classify each node into one of C categories.

2 Method

Similar to CNNs, GCNs learn new feature representations of X over multiple layers. In kth-layer, the node representation is a matrix $H^{(k)} \in \mathbb{R}^{n \times d}$. The core idea of GCN is the node representation of a node i at kth-layer depends on the node representations of all its neighboring nodes at $(k-1)$-th layer. This is called **feature propagation**. After stacking L layers, the final node representations of node i may contain information propagated from all neighboring nodes which are at most $k-$hop away from i.

3 Formula

3.1 Matrix form

- In one line:

$$X = H^{(0)} \to H^{(1)} \to ... \to H^{L} \to \hat{Y} \tag{1}$$

- Matrix form:

$$H^{(l+1)} = \sigma(\tilde{D}^{-0.5}\tilde{A}\tilde{D}^{-0.5}H^{(l)}W^{(l)}) \tag{2}$$

$$\hat{Y} = \text{softmax}(H^{(L)}) \tag{3}$$

  where $H^{(l)} \in \mathbb{R}^{n \times d}$, $\tilde{A} \in \mathbb{R}^{n \times n}$, $\tilde{D} \in \mathbb{R}^{n \times n}$, $W^{(l)} \in \mathbb{R}^{d \times d}$.

- Step-by-step:

  Let $S = \tilde{D}^{-0.5}\tilde{A}\tilde{D}^{-0.5}$.

  - Feature propagation:

$$\bar{H}^{(k)} = SH^{(k-1)} \tag{4}$$

  - Feature transformation

$$H^{(k)} = \text{ReLU}(\bar{H}^{(k)}\Theta^{(k)}) \tag{5}$$

– Classifier

$$\hat{Y} = \text{softmax}(H^{(L)})$$

4 Experimental result and Takeaways

Run semi-supervised node classification on Citeseer, Cora, Pubmed, NELL. Achieved state-of-the-art performance at that time.

4.1 Some side results

- Renormalization trick made huge improvement over existing methods (like Chebyshev filter)
- Best results are obtained with 2-layer or 3-layer model. Beyond that, performance drops. By adding residual connection, deeper model can achieve nearly the same performance with shallow ones.

Part II

Simplifying Graph Convolutional Network (SGCN)

Paper

1 Problem

Given dataset $X \in \mathbb{R}^{n \times d}$ where n is the number of nodes, d is the number of features. We want to classify each node into one of C categories.

2 Method

Similar to GCN, but drop feature transformation step.

3 Formula

In one line:

$$\hat{Y} = \text{softmax}(S^K X \Theta) \tag{6}$$

4 Experimental result

Achieve comparable performance with GCN. Can be run on larger dataset (Reddit). Memory requirement and running time are better than GCN by constant factors.

Part III

Graph Attention Network (GAT)

Paper

1 Problem

Given dataset $X \in \mathbb{R}^{n \times d}$ where n is the number of nodes, d is the number of features. We want to classify each node into one of C categories.

2 Method

Learn new feature representations of X over multiple layers. Input to a layer is a set of node features $h = \{h_1, h_2, .., h_N\}$ where $h_i \in R^F$ and F is the number of features in each node. Output of that layer is a set of node features $h' = \{h'_1, h'_2, .., h'_N\}$ where $h'_i \in R^{F'}$ and F' is the number of features in each node. Information is propagated among all neighboring nodes via **attention mechanism**. Key point of the attention mechanism is that it allows for assigning different importances to nodes in the same neighborhood.

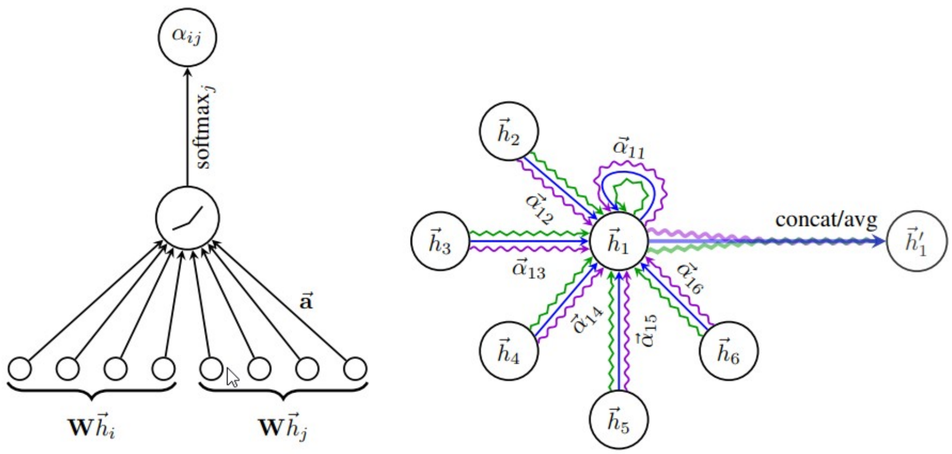

Figure 1: **Left:** The attention mechanism $a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$ employed by our model, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation. **Right:** An illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\vec{h}'_1$.

3 Formula

- Attention of node i to node j:

$$e_{ij} = a(Wh_i, Wh_j) \tag{7}$$

- Attention, normalized:

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k \in N_i} exp(e_{ik})} \tag{8}$$

- Attention, normalized, fully expanded equation:

$$\alpha_{ij} = \frac{exp\Big(\text{LeakyReLU}\big(a^T[Wh_i||Wh_j]\big)\Big)}{\sum_{k \in N_i} exp\Big(\text{LeakyReLU}(a^T[Wh_i||Wh_k])\Big)} \tag{9}$$

  where $||$ is concatenation and $a \in \mathbb{R}^{2F}$

- Feature propagation:

$$h_i^{'} = \sigma\left(\sum_{j \in N_i} \alpha_{ij} W h_j\right) \tag{10}$$

- Feature propagation, with multi-head attention:

$$h_i^{'} = ||_{k=1}^{K} \sigma\left(\sum_{j \in N_i} \alpha_{ij}^k W^k h_j\right) \tag{11}$$

- Last layer:
$$h_i' = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in N_i}\alpha_{ij}^k W^k h_j\right) \tag{12}$$

4 Experimental results and Takeaways

- Run transductive learning semi-supervised node classification on Citeseer, Cora, Pubmed. Achieve significantly better result than GCN.
- Run inductive learning node classification on PPI. Achieve state-of-the-art and significantly better result than GraphSAGE-GCN.

Part IV
LightGCN: Simplifying and Powering GCN for Recommendation

Paper

1 Problem

Given N users, M items and a list $\{(u,i)\}$ of interactions between user u and item i, predict whether a user will interact with an item.

2 Method

- Learn **latent features (embedding)** to represent users and items, then perform prediction based on the embedding vectors
- Problem is represented as bipartite graph, one side are users and the other side are items.
- Adopt the feature propagation idea of GCN, however abandon the use of feature transformation and nonlinear activation. The kth-layer representation of a user u contain information propagated from all the nodes which are k-hop away from u. Final representation is a weighted sum of all the layer-presentations.
- Model is trained with Bayesian Personalized Ranking (BPR) loss.

3 Formula

3.1 Matrix form

- Adjacency matrix:
$$A = \begin{pmatrix} 0 & R \\ R^T & 0 \end{pmatrix} \tag{13}$$
where $R_{ui} = 1$[user u interacted with item i].

- Feature propagation:
$$E^{(k+1)} = (D^{-0.5}AD^{-0.5})E^{(k)} \tag{14}$$

- Final representation:
$$E = \alpha_0 E^{(0)} + ... + \alpha_K \tilde{A}^K E^{(K)} \tag{15}$$
where $\tilde{A} = D^{-0.5}AD^{-0.5}$

- Prediction:
$$\hat{y}_{ui} = e_u^T e_i \tag{16}$$

- Loss function:
$$L_{BPR} = -\sum_{u=1}^{M} \sum_{i \in N_u} \sum_{j \notin N_u} \ln\sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda ||E^{(0)}||^2 \tag{17}$$

4 Experimental Results and Takeaways

- Run on Gowalla, Yelp2018, Amazon-Book which are 3 datasets with the number of users 30,000 - 50,000; number of items 40,000 - 90,000; number of edges 1,00,000 - 3,000,000. Outperforms NGCF (which was state-of-the-art at collaborative filtering) by a large margin.

- The author argued that much of the effectiveness of LGCN was due to its successful embedding smoothness (i.e. two users overlapping in a large number of items should have similar embedding vectors).

Part V

Graph Neural Networks for Social Recommendation

Paper

1 Problem

Given $U = \{u_1, u_2, ..., u_n\}$ and $V = \{v_1, v_2, ..., v_m\}$ the set of users and items respectively; $R \in \mathbb{R}^{m \times n}$ the user-item rating matrix which have two parts: observed part $\mathbb{O} = \{(u_i, v_j) | r_{ij} \neq 0\}$ and missing part $\mathbb{T} = \{(u_i, v_j) | r_{ij} = 0\}$; and user-user social graph $T \in \mathbb{R}^{n \times n}$. We are to predict the missing part of R.

2 Method

- Learn user and item embeddings, use these embeddings to perform prediction
- User latent factors are obtained by combining information from both item space and item space
- Item latent factors are obtained via user aggregation
- Loss function is simply defined as discrepancy between predicted and ground-truth ratings for ratings in observed set

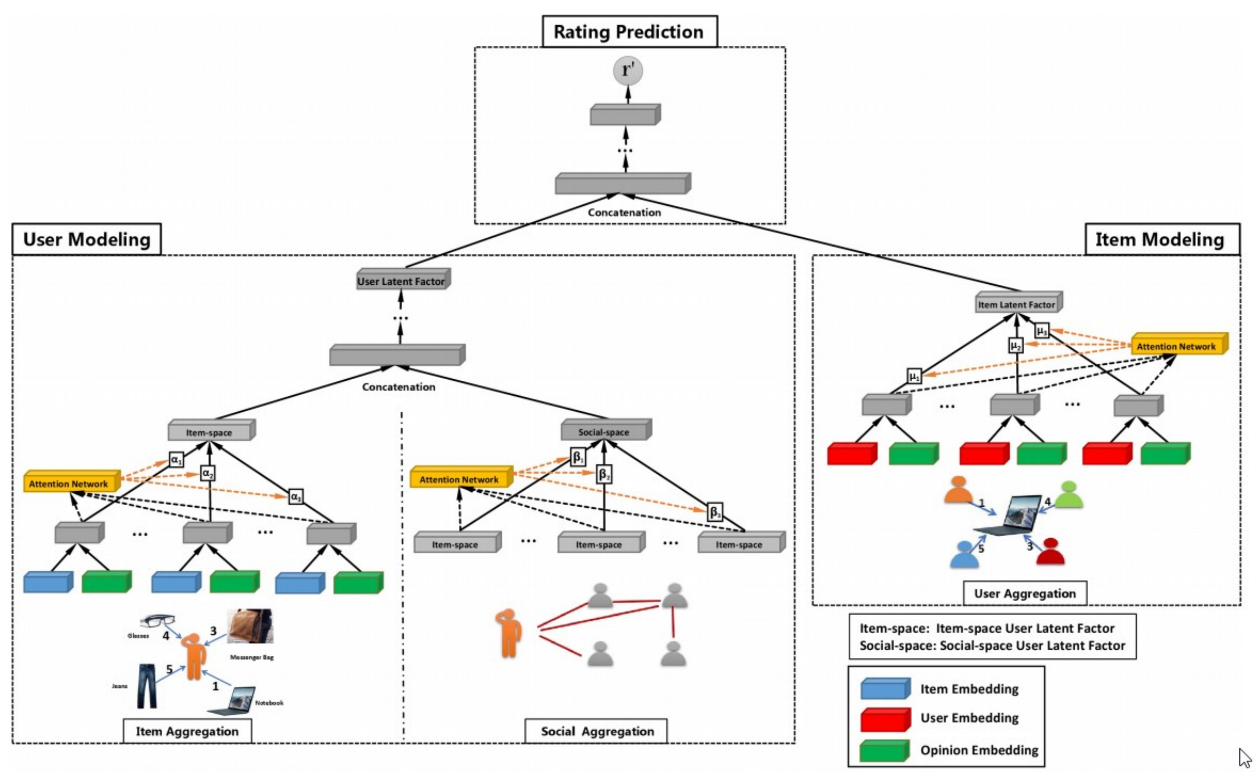

Figure 2: The overall architecture of the proposed model. It contains three major components: user modeling, item modeling, and rating prediction.

3 Formula

3.1 User Modeling

User modeling aims to learn user latent factors, denoted as $h_i \in R^d$ for user u_i. To do that, we try to learn item-space user latent factor $h_i^I \in R^d$ from the user-item graph and user-space user latent factor $h_i^S \in R^d$, then combine them together to obtain h_i.

3.1.1 Item Aggregation

- Item-space user latent factor:

$$h_i^I = \sigma\Big(W\Big\{\sum_{a \in C(i)} \alpha_{ia} x_{ia}\Big\} + b\Big) \tag{18}$$

  where x_{ia} is opinion-aware interation between user i and item a, $C(i)$ is set of all items rated by user i, α_{ia} the attention weight of the iteration between user i and item a.

- Opinion-aware interaction:

$$x_{ia} = g_v\Big([q_a||e_r]\Big) \tag{19}$$

  where g_v is a MLP, q_a is learnable item embedding of item a, e_r is the embedded rating of rating r where $r \in \{1, 2, 3, 4, 5\}$.

- Attention weight α_{ia}:

$$\alpha_{ia}^* = w_2^T \sigma\Big(W_1[x_{ia}||p_i] + b_1\Big) + b_2 \tag{20}$$

$$\alpha_{ia} = \text{softmax}(\alpha_{ia}^*) \tag{21}$$

  where p_i is learnable user embedding of user i.

3.1.2 Social Aggregation

- Social-space user latent factor:

$$h_i^S = \sigma\Big(W\Big\{\sum_{o \in N(i)} \beta_{io} h_o^I\Big\} + b\Big) \tag{22}$$

  where β_{io} is the strength of connection between user i and user o

- Social attention β_{io}

$$\beta_{io}^* = w_2^T \sigma\Big(W_1[h_o^I||p_i] + b_1\Big) + b_2 \tag{23}$$

$$\beta_{io} = \text{softmax}(\beta_{io}^*) \tag{24}$$

  where p_i is learnable user embedding of user i.

3.1.3 Learning User Latent Factor

$$h_i = FC^l\Big([h_i^I||h_i^S]\Big) \tag{25}$$

where FC is one fully connected layer of a neural network.

3.2 Item Modeling

3.2.1 User Aggregation

- Item latent factor:

$$z_j = \sigma\bigg(W\Big\{\sum_{t\in B(j)} \mu_{jt} f_{jt}\Big\} + b\bigg) \tag{26}$$

  where $B(j)$ is the set of users who interacted with item j, f_{jt} the opinion-aware interaction representation of user t for item j, and μ_{jt} the user attention of user t in contributing to item j.

- User attention to item:

$$\mu_{jt}^* = w_2^T\sigma\bigg(W_1[f_{jt}||q_j] + b_1\bigg) + b_2 \tag{27}$$

$$\mu_{jt} = \text{softmax}(\mu_{jt}^*) \tag{28}$$

- Opinion-aware interaction:

$$f_{jt} = g_u([p_t||e_r]) \tag{29}$$

  where g_u is a MLP, p_t is a learnable user embedding.

3.3 Rating prediction

$$r'_{ij} = w^T\text{FC}^{l-1}\bigg([h_i||z_j]\bigg) \tag{30}$$

where FC is one fully connected layer of a neural network.

3.4 Loss function

$$Loss = \frac{1}{2|\mathbb{O}|}\sum_{i,j\in\mathbb{O}}(r_{ij}^{'} - r_{ij})^2 \tag{31}$$

4 Experiment Results and Takeaways

- Use two rating datasets: Ciao and Epinions
- Train-Validation-Test split: (x, (1- x)/2, (1 - x)/2), $x \in [60\%, 80\%]$. Achieve state-of-the-art for both datasets.
- Attention mechanism is an important factor in the sucess of this model.