Recommender Systems with GNN

- User: \( u \), \( u' \)
- Item
  - Items interacted by both users \( u \) and \( u' \)
  - Likely edges

Similar users:

Slides from Jure Leskovec
Recommender systems

• Information explosion in the era of the Internet
  • 10K+ movies in Netflix
  • 12M products in Amazon
  • 70M+ music tracks in Spotify
  • 10B+ videos on YouTube
  • 200B+ pins (images) in Pinterest

• Personalized recommendation helps users effectively explore content of interest
Bipartite graph

- Recommender systems can be naturally modelled as a bipartite graph
  - Nodes: users and items
  - Edges: interaction between users and items
    - Clicks, purchases, reviews etc.
    - Often have a timestamp
Task

• Given past user-item interactions, predict new items a user will interact with
  • Can be case as link prediction: predict new interaction given past edges
Top-K recommendations

• Recommend K items for each user
  • K needs to be low—is for the recommendations to make sense
  • Typically 10-100

• Goal: as many positive items as possible to be in the top-K recommendations
  • Positive items: items that the user will interact with in the future

• Evaluation metric: Recall@K
Recall@K

- For each user $u$,
  - Let $P_u$ be a set of positive items the user will interact with in the future
  - Let $R_u$ be the set of items recommended by the model
    - In top-K recommendation, $|R_u| = K$
    - Items the user has already interacted with are excluded

- Recall@K for user $u$ is $|P_u \cap R_u| / |P_u|$
- Recall@K for the dataset is the averaged Recall@K across users
Score function

- To get the top-K items, define a score function for user item interaction
  - For $u \in U, v \in V$, we need to get a real-valued scalar $score(u, v)$
  - K items with the largest scores for a given user $u$ (excluding already-interacted items) are then recommended

For $K = 2$, recommended items
**Embedding-Based Models**

- Embedding-based models for scoring user-item interactions:
  - Compute embeddings $z_u \in \mathbb{R}^d$ and $z_v \in \mathbb{R}^d$ for pairs $(u, v)$ of users and items
  - Define $f_\theta(\cdot, \cdot): \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ to be a parameterized function
  - $score(u, v) = f_\theta(z_u, z_v)$
Training objective

• Optimize $\{z_u\}_{u \in U}, \{z_v\}_{v \in V}, \theta$ to achieve high recall@K on seen (i.e., training) user-item interactions

• Hope that would lead to high recall@K on unseen (i.e., test interactions)
Surrogate Loss functions

• recall@K is not a differentiable function, so cannot apply gradient-based optimization

• User *surrogate functions* to enable gradient based optimization
  • Binary loss
  • Bayesian Personalized Ranking (BPR) loss

• Surrogate losses should be
  • Differentiable
  • Aligned with the original training objective
Binary loss

• Sum over the positive edges $E$ and negative edges $E_{neg}$
  • Negative edges are those that are absent in the training set

\[- \frac{1}{|E|} \sum_{(u,v) \in E} \log(\sigma(f_\theta(u,v))) - \frac{1}{|E_{neg}|} \sum_{(u,v) \in E_{neg}} \log(1 - \sigma(f_\theta(u,v)))\]

During training, these terms can be approximated using mini-batch of positive/negative edges

• Make $f_\theta(u,v)$ high for observed edges $(u, v)$, and low otherwise
• Aligns with recall@K in the sense that we are scoring positive edges higher
Binary loss: issue

- Binary loss pushes *all* negative edges down and *all* positive edges up
- But we can imagine a perfect recommender where some negative edges are higher than some positive edges
  - Only need for positive edges to be higher than negative edges for each particular user

Perfect recall@K, but high binary loss
BPR loss

• Bayesian Personalized Ranking (BPR) loss is a personalized surrogate loss that aligns better with the recall@K metric

• For each user $u^* \in U$, define the rooted positive/negative edges as
  
  • Positive edges: $E(u^*) = \{(u^*, v) | (u^*, v) \in E\}$
  
  • Negative edges: $E_{neg}(u^*) = \{(u^*, v) | (u^*, v) \in E_{neg}\}$

Note: The term “Bayesian” is not essential to the loss definition. The original paper [Rendle et al. 2009] considers the Bayesian prior over parameters (essentially acts as a parameter regularization), which we omit here.
Training objective: BPR loss

• For each user $u^*$, we want the scores of rooted positive edges $E(u^*)$ to be higher than those of rooted negative edges $E_{neg}(u^*)$
  • Aligns with what we want from maximizing recall@k
• BPR loss for user $u^*$:

\[
\text{Encouraged to be positive for each user} \\
\text{positive edge score is higher than negative edge score}
\]

\[
\text{Loss}(u^*) = \frac{1}{|E| \cdot |E_{neg}|} \sum_{(u^*, v_{pos}) \in E(u^*)} \sum_{(u^*, v_{neg}) \in E_{neg}(u^*)} - \log \left( \sigma \left( f_\theta(u^*, v_{pos}) - f_\theta(u^*, v_{neg}) \right) \right)
\]

Can be approximated using mini-batch

• Final BPR Loss:

\[
\frac{1}{|U|} \sum_{u^* \in U} \text{Loss}(u^*)
\]
BPR loss: mini-batch training

• Sample a subset of users $U_{\text{mini}} \subset U$
  • For each user $u^* \in U_{\text{mini}}$, we sample one positive item $v_{\text{pos}}$ and a set of sampled negative items $V_{\text{neg}}$

• Mini-batch loss:

$$\frac{1}{|U_{\text{mini}}|} \sum_{u^* \in U_{\text{mini}}} \frac{1}{|V_{\text{neg}}|} \sum_{v_{\text{neg}} \in V_{\text{neg}}} -\log \left( \sigma \left( f_{\theta}(u^*, v_{\text{pos}}) \right) - \sigma \left( f_{\theta}(u^*, v_{\text{neg}}) \right) \right)$$

Average over users in the mini-batch
Embedding Models for Recommender Systems

• Underlying idea: “collaborative filtering”
  • Recommend items for a user by collecting preferences of many other similar users
  • Similar users tend to prefer similar items
• Graph embeddings capture similarity between users/items
Embedding Models for Recommender Systems

• Embedding-based models can capture similarity of users/items!
  • Low-dimensional embeddings cannot simply memorize all user-item interaction data
  • Embeddings are forced to capture similarity between users/items to fit the data
  • This allows the models to make effective prediction on unseen user-item interactions
Neural Graph Collaborative Filtering

• Explicitly incorporates high-order graph structure (i.e., neighbourhood information rather than just edges) when generating user/item embeddings

• Key idea: use a GNN to generate graph-aware user/item embeddings
NGCF Framework

• Start with a user-item bipartite graph

• NGCF framework:
  • Prepare shallow learnable embedding for each node
  • User multi-layer GNNs to propagate embeddings along the bipartite graph
    • Capture higher-order structure
  • Final embeddings are explicitly graph-aware

• Jointly learn:
  • Shallow user/item embeddings
  • GNN’s parameters
NGCF learning

• Initialize shallow embeddings
• Iteratively update node embedding using neighbouring embeddings

\[ h_v^{(k+1)} = \text{COMBINE}\left(h_v^{(k)}, \text{AGGR}\left(\{h_u^{(k)}\}_{u \in N(v)}\right)\right) \]

\[ h_u^{(k+1)} = \text{COMBINE}\left(h_u^{(k)}, \text{AGGR}\left(\{h_v^{(k)}\}_{v \in N(u)}\right)\right) \]

• AGGR() can be MEAN(), COMBINE(x, y) can be ReLU(Linear(Contact(x, y)))
NGCF learning

• After K rounds of aggregation, get final user/item embeddings $h_u^{(K)}$ and $h_v^{(k)}$

• $score(u, v) = h_u^{(K)} \cdot h_v^{(k)}$

• Can now compute BPR and backpropagate
PinSAGE: Scaling Up NGCF

• Data: Pinterest pins
• Pin embedding unifies visual, textual, and graph information
• Embeddings for new content available within seconds
PinSAGE: Scaling Up

- Shared negative samples across users in a mini-batch
- Mining for hard negative samples
- Curriculum learning
- Mini-batch training of GNNs on a large graph
Shared negative samples

• In BPR loss, we sampled a set of negative edges for each positive edge
• Costs $O(|U_{\text{mini}}||V_{\text{neg}}|)$ to sample and compute the loss for the negative edges
• Idea: sample $V_{\text{neg}} = \{v_{\text{neg}}\}$ across all users in the minibatch $U_{\text{mini}}$
  • Only compute $|V_{\text{neg}}|$ embeddings
Curriculum learning

- Idea: make the negative samples *gradually harder* in the process of training
- At the n-th epoch, add n-1 hard negative items
- The model will gradually learn to make finer-grained predictions
Curriculum learning II

- Idea: use harder and hard negative samples
Hard negatives

• Most negatives that are sampled are “easy negatives”

• Hard negatives are nodes that are close (but not connected) to the user node in the graph

• Obtain hard negatives for u:
  • Compute Personalized Page Rank (PPR) for user u
    • PPR is the probability of v occurs on a random walk starting at u
    • Sample item notes that are ranked high but not too high by PPR to U
      • Item nodes that are close not connected to user node