Movie Recommendation System

CS-725 Foundations of Machine Learning

Smita Gautam(24M0845), Deeksha Koul(24D0365),
Sagar Singh(24D0367), Mohit Singh Tomar(24D0372)

November 25, 2024

Introduction

- Developing recommendation systems that filter *Relevant information* for the user is critical.
- We explored the potential of Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs) for movie recommendation on the MovieLens100K dataset.
- GCNs were chosen as they are able to exploit both i.e: the information present in user/item features and their interaction graph.
- We also studied locality sensitive hashing for faster inference.

MovieLens Dataset

- torch_geometric.datasets.MovieLens100K

Node type	#nodes	#features
User	943	24
Movie	1,682	18
User-Movie	80, 000	

- 80/10/10 train-val-test split on user nodes
- Edges with rating >= 3 are preserved in the graph

$$h_v^{(0)} = x_v$$

$$h_v^{(k+1)} = \sigma\left(\mathbf{W}_k \sum_{u \in \mathcal{N}(v)} \frac{h_u^{(k)}}{|\mathcal{N}(v)|} + \mathbf{B}_k h_v^{(k)}\right), \quad \forall k \in \{0, \ldots, K-1\}$$

$$z_v = h_v^{(K)}$$

K: Number of layers, σ : *Activation function*,
$h_v^{(0)}$: *Initial node embedding*,
$h_v^{(k)}$: *Embedding at kth layer*,
z_v : *Embedding at final layer*,
W_k : *Weight matrix for neighborhood aggregation*,
B_k : *Weight matrix for self transformation*

Model Architecture(GAT)

$$h_v^{(k)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_{vu} \mathbf{W}^{(k)} h_u^{(k-1)}\right)$$

$$e_{vu} = a\left(\mathbf{W}^{(k)} h_u^{(k-1)}, \mathbf{W}^{(k)} h_v^{(k-1)}\right)$$

$$\alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{i \in \mathcal{N}(v)} \exp(e_{vi})}$$

$\mathcal{N}(v)$: Neighborhood of a node including self node
a: single layer feed-forward neural network.

Loss Function

Bayesian Personalized Loss (BPR):

$$\mathscr{L}_{\text{BPR}} = -\sum_{u=1}^{M} \sum_{i \in N_u} \sum_{j \notin N_u} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj})$$

Max Margin Loss:

$$\mathscr{L}_{\text{Margin}} = -\sum_{u=1}^{M} \sum_{i \in N_u} \sum_{j \notin N_u} max\{0, \Delta + \hat{y}_{uj} - \hat{y}_{ui}\}$$

Model Hyperparameters

Hyperparameter	**Value**
Number of layers	2
Hidden layer dimension	128
Output layer dimension	64
Negative samples	10
Loss margin	0.01
Learning rate	0.001
Epochs	15
Top-K	10

Evaluation results

MoveLens100K	P@10	R@10	MRR	NDCG@10
ALS	0.0375	0.0178	0.2083	0.0010
SVD	0.1695	0.0162	0.5946	0.1472
NCF	0.067	0.008	0.41	0.069
GCN + Margin	0.2947	0.0485	0.638	0.3088
GCN + BPR	0.3010	0.0512	0.660	0.312
GCN + RP_3*	0.2568	0.04318	0.618	0.2910
GAT + Margin	0.3182	0.0581	0.679	0.3221
GAT + BPR	0.3481	0.0597	0.690	0.3418

Random LSH Inferencing

- Create hyperplanes to split datapoints and assign a value of either 0 or 1 for every hyperplane.
- Generate hash codes for both users and movies.
- Bucketize the movie hash codes. Number of buckets, $N = 2^{\#planes}$
- For every user, find the closest bucket using hamming distance. The movies in that bucket are our candidate vectors.
- Do the same evaluation on these candidates.
- Observed a faster inference time, in GCN model, the evaluation is done at rate of ≈ 40 users per second. In random projection this gets increased to ≈ 150 users per second.

Conclusion

- Leveraged GCN and GAT models on MovieLens100K dataset
- Performance comparison of Graph models with baselines as SVD, ALS and NCF
- Implemented two loss functions i.e BPR and max-margin for training graph networks.
- Implemented random projection(Random LSH) technique to optimize the inference process.
- Demo of our trained GCN Model built using Gradio, and deployed on Huggingface. Demo Link

Team Contributions

- Problem statement: Mohit
- Baselines implementation of SVD, ALS: Smita
- Baselines implementation of NCF: Sagar
- Evaluation and loss implementation: Deeksha
- Model architecture: Mohit
- Demo on Huggingface: Smita
- Presentation: all

References

[1] Thomas N. Kipf, Max Welling, “Semi-Supervised Classification with Graph Convolutional Networks,” *ICLR 2017*.

[2] P. Velikovi, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph Attention Networks,”

[3] Stanford University, “Graph Neural Network Reference,” `https://web.stanford.edu/class/cs224w/`

[4] Pinecone, “Locality Sensitive Hashing and Random Projection,” `https://www.pinecone.io/learn/series/faiss/locality-sensitive-hashing-random-projection/`