Implicit Rating Prediction

- MSDS630 Melvin Vellera Haotian Gong

Overview

Personalized recommendation is the key to attract and retain users. By using an implicit feedback recommendation system, we can deduce user preferences even without their explicit input and improve recommendations for them.

Our training set has 970K rows with four columns, and each row represents a user liking a specific item, with a corresponding item and context feature. After EDA and pre-processing, we fed data into various models to learn an implicit user rating. We predicted implicit ratings for 381K user & item cases.

EDA

1. User IDs and Item IDs have the same range in training and test set.

2. About 80% users and 14% of items are new in the test set. To deal with this, we used mean embedding for new users and items.

3. About 64% users in the training set are under only one specific context. So randomly sampling context feature might not be reasonable.

4. We defined item features which only have less than 30% overlapping unique users as “unpopular” item features. Only 30% users in the training set “like” these item features. So random sampling item features should be OK.

Techniques - Negative Sampling

- **Negative Item Sampling**

  For each user in the training set, we negative sampled random and non-repetitive items.

- **Item Sampling Ratio**

  The sampling rate of item is inverse related to item popularity. We used four tiers of ratio: ✕2, ✕4, ✕5, ✕6, negatively correspond to each quartile.

- **What else ?**

  We kept the context feature for sampled user. And item features were added by merging item table. Deleted duplicate training data.

Techniques - Unknown User Embedding

Mean Embedding

After the model has been trained completely, we take the mean of all of the user and item embeddings and assign it as an "extra" embedding at the end which gets assigned to any unknown user or item for the test set

Unknown Embedding Imputation

We use a new user and item embedding to randomly replace some users and items during training such that by the end of training, this new user and item embedding represents all users and items (similar to mean embedding)

Techniques - Models

Matrix Factorization

Matrix factorization models with user-item, user-item with bias, and user-item with bias and additional features were considered

Random Forest

We extracted user and item embeddings from the matrix factorization model, added additional features and trained a Random Forest model

Neural Networks

Concatenated embeddings for all features were passed through a multi-layer neural network with ReLU and dropout

Experimental Results - MF Model

Model	Features	Embedding Size	Epochs	Train Loss	Public Loss
MF w/ Cross Validation	Users+Items	50+50	7, 5 folds	0.168	0.454
MF	Users+Items+Additional Features	50+50+50+50	35	0.187	0.481
MF	Users+Items (with bias)	50+50	35	0.166	0.432

- Learning Rate: 0.1
- Weight Decay: 1e-6

Experimental Results - Random Forest

Model	Features	Number of Trees	Min Samples Split	Max Features	Train Loss (MF)	Train Loss (RF)	Public Loss
Random Forest	Users & Item Embeddings, Item Feature, Context Feature	100	2	Sqrt	0.189	0.195	0.7

Note: As the public loss was not encouraging, we did not experiment further with the Random Forest

Experimental Results - Neural Networks

Sampling Strategy	Total Embedding Size	Layers	Epochs	Batch Size	Learning Rate	Weight Decay	Dropout	Train Loss	Public Loss
Popularity	120	4	30	All data	0.01	0	0.1	0.346	0.479
Popularity	100	3	35	All data	0.1	1e-6	0.1	0.246	0.453
Random	200	2	15	50000	0.01	1e-3	0.8	0.335	0.424
Popularity	200	2	30	50000	0.01 (15), 0.001 (15)	1e-3	0.2	0.295	0.415
Popularity	280	2	30	50000	0.01 (15), 0.001 (15)	1e-3	0.2	0.291	0.409

Experimental Results - Average Best Models

MF Model Weight	NN Model Weight	Public Loss
1	1	0.411
1	2	0.408
1	3	0.40735
1	4	0.40711
1	5	0.40710

Lessons Learned

1. EDA is the key to understanding the data and generating pre-processing methods.

2. To better train an MF model with implicit ratings, we should try to mimic the real-word scenario by optimizing negative sampling

3. Mean embedding for unknown users can improve performance

4. Neural networks need extensive hyperparameter tuning in order to get good performance

Thank you