Assignment3

12112124 尹晨

The purpose of the task is to reproduce the experiment of the paper to a certain extent, and use the neural network model to train and evaluate the data of the movie recommendation system. Three models are implemented: GMF (Generalized Matrix Factorization), MLP (Multi-Layer Perceptron) and NeuMF (Neural Matrix Factorization), and their performance was evaluated.

First, according to the task requirements, the latest version of the MovieLens dataset (latest) was downloaded from the official website for this task. Considering the educational significance of task repetition and the limited local resources, this dataset is also relatively small.

Main modules and functions

1. **Data loading and preprocessing**

- Use 'pandas' to read MovieLens data set.
- Convert the Dataset to PyTorch Dataset via the MovieLensDataset class and load the training and test data via DataLoader.

2. **Model definition**

- **GMF**: The score is calculated by the product of the elements of the embedding vector of the user and the item.
- **MLP**: Score is calculated through a multi-layer fully connected network, connected using embedded vectors of users and items as input.
- **NeuMF**: Combines the benefits of GMF and MLP, linking the outputs of both to calculate scores through a fully connected layer.

3. **Model training**

- Use the binary cross entropy loss function (' nn.BCELoss ') and the Adam optimizer (' torch.optim.Adam ').
- Complete training data traversal per epoch, calculating and backpropagating errors to update model parameters.

4. **Model evaluation**

- Use hit ratio (HR) and normalized cumulative loss gain (NDCG) as evaluation metrics.
- evaluate_model calculates the user's first k recommended items, compares these items with actual favorites, calculates HR and NDCG.
- 'train_and_evaluate_mlp_models' function is used to train and evaluate MLP models with different levels and dimensions.

5. **Result visualization**

- Use 'matplotlib' to visualize HR and NDCG scores of different models in different dimensions.

Concepts:

HR and NDCG Calculation Formulas

Hit Rate (HR) Calculation Formula

Hit Rate (HR) measures the proportion of recommended items in the top (k) that are actually liked by the user. The formula is:

$$HR@k = \frac{\sum_{u \in U} \delta(\text{Hit@k}(u))}{|U|}$$

where:

- U represents the set of users.
- $\delta(\text{Hit@k}(u))$ is 1 if the user's top (k) recommended items contain at least one item that the user actually likes, otherwise 0.
- $\text{Hit@k}(u)$ is the set of the top (k) recommended items for user (u).

Normalized Discounted Cumulative Gain (NDCG) Calculation Formula

NDCG measures the relevance of the recommended items, taking into account the order of the recommendations. The formula is:

$$NDCG@k = \frac{DCG@k}{IDCG@k}$$

where:

- $DCG@k$ is the Discounted Cumulative Gain for the top (k) recommended items, calculated as:

$$DCG@k = \sum_{i=1}^{k} \frac{rel_i}{\log_2(i+1)}$$

Here, rel_i is the relevance score of the (i)-th recommended item (1 if the item is actually liked by the user, otherwise 0).

- $IDCG@k$

  is the Ideal Discounted Cumulative Gain, which is the

  $$DCG@k$$

  for an ideal ranking of items, sorted by relevance in descending order:

$$IDCG@k = \sum_{i=1}^{|REL_k|} \frac{1}{\log_2(i+1)}$$

Here,

$$|REL_k|$$

is the total number of relevant items, capped at (k).

Using these formulas, we can calculate HR and NDCG. HR focuses on the accuracy of the recommendations, while NDCG also considers the ranking order of the recommended items.

Result visualization

3 different models

After 200 training sessions, the output test results are:

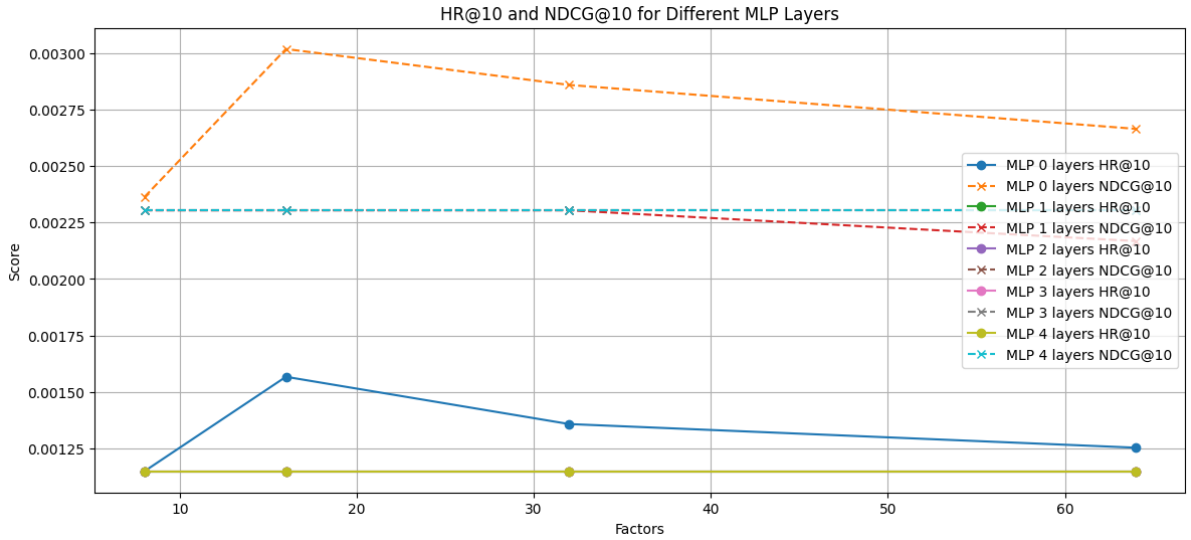
```
GMF Model - HR@10: 0.0011495454070435782, NDCG@10: 0.0023037725269350736
MLP Model - HR@10: 0.0011495454070435782, NDCG@10: 0.0020722311558457785
NeuMF Model - HR@10: 0.0011495454070435782, NDCG@10: 0.00268324403472433
```

MLP with different layers

After 200 training sessions, the output test results are:

```
          HR@10                                                        \
   MLP 0 layers MLP 1 layers MLP 2 layers MLP 3 layers MLP 4 layers
8      0.001150      0.00115      0.00115      0.00115      0.00115
16     0.001568      0.00115      0.00115      0.00115      0.00115
32     0.001359      0.00115      0.00115      0.00115      0.00115
64     0.001254      0.00115      0.00115      0.00115      0.00115

         NDCG@10
   MLP 0 layers MLP 1 layers MLP 2 layers MLP 3 layers MLP 4 layers
8      0.002362     0.002304     0.002304     0.002304     0.002304
16     0.003017     0.002304     0.002304     0.002304     0.002304
32     0.002858     0.002304     0.002304     0.002304     0.002304
64     0.002664     0.002168     0.002304     0.002304     0.002304
```



Analysis

The above output is reasonable. The HR (Hit Rate) and NDCG (Normalized Discounted Cumulative Gain) metrics of the model usually do not reach very high values, especially in the benchmark test of the recommendation system. Here's an explanation of the results:

1. **Hit Rate (HR)@10** : indicates the proportion of items that actual users like among the top 10 recommended items.
   In the comparison results of different models, the value of HR@10 is low, which indicates that the recommendation effect of the model needs to be improved. Recommendation systems are usually faced with a large number of candidates, and it is not easy to hit the user's favorite project.
   In the MLP multilayer test, the MLP_0_layers model performs best at factor16.
2. **NDCG@10** : Considering the order and relevance of recommended items, the cumulative gain of discount after standardization. In the results, NDCG@10 also has a low value, which indicates that the ranking quality of recommended items needs to be improved.
   In the MLP multilayer test, the MLP_0_layers model performs best at factor16.

These results show that the recommendation performance of the model is not ideal under the current data and parameter Settings. Here are some possible directions for improvement:

1. **Data preprocessing** : More cleaning and processing of data may be required.
2. **Model optimization** : Try different hyperparameters, architectures or optimization algorithms to improve model performance.
3. **Feature Engineering** : Introduce more user and project features.
4. **Training data** : Increasing the size and diversity of training data may improve the generalization ability of the model.

As this task is only a repetition process, and local resources are limited, no detailed updates will be made.

Summary

The code attempts to replicate a complete recommendation system model training and evaluation process, from data loading and preprocessing, to model definition, training, and evaluation, to the final result visualization. Through the comparison of different models, the optimal recommendation system architecture can be found to provide more accurate recommendation results.