Sadi Toirov (s.toirov@innopolis.university)

1. Introduction

The recommendation system analyzes the past preferences of the user concerned, and then it uses this information to try to find similar movies

2. Data analysis

- Dataset consist of 100k ratings from 943 users on 1682 movies.
- Ratings are ranged from 1 to 5
- Each user has rated at least 20 movies
- Dataset consist of ratings, user info and movies info.
    - User info: age, gender, occupation, zip code
    - Movie info: movie title, release date, IMDb URL and 19 genres.
- For training model used following data:
    - user_id, movie_id and 19 genres (one hot)

3. Model Implementation

```python
class HybridModel(nn.Module):
    """HybribModel
        - take user_id, item_id and some item_features
        - returns predicted rating

    """

    def __init__(self, num_users, num_items, embed_dim, items_features, hidden_dim, dropout=0.1):
        super(HybridModel, self).__init__()

        self.user_embedding = nn.Embedding(num_users, embed_dim)
        self.item_embedding = nn.Embedding(num_items, embed_dim)

        self.fc_out = nn.Sequential(
            nn.Linear(2 * embed_dim + items_features, hidden_dim),
            nn.ReLU(),
            nn.Dropout(dropout),

            nn.Linear(hidden_dim, hidden_dim // 2),
            nn.ReLU(),
            nn.Dropout(dropout),

            nn.Linear(hidden_dim // 2, 1),
        )
        self.dropout = nn.Dropout(dropout)

    def forward(self, user_id, item_id, item_feat):
        user_embed = self.dropout(self.user_embedding(user_id))
        item_embed = self.dropout(self.item_embedding(item_id))

        x = torch.cat((user_embed, item_embed, item_feat), dim=1)

        x = self.fc_out(x)

        return x
```

Model Architecture:

- user_embedding: nn.Embedding (embed_dim=64)
- item_embedding: nn.Embedding (embed_dim=64)
- dropout: 0.3
- fc_out: linear layer which outputs the rating of the movie

4. Model Advantages and Disadvantages

4.1 Advantages

1. **Incorporation of item features**: By including additional item features alongside user and item embeddings, your model can capture more complex relationships and dependencies that might exist beyond user-item interactions. This can lead to a richer representation of items and potentially better recommendations.
2. **Customizable architecture**: The use of fully connected layers allows for flexibility in model architecture. You can adjust the number of hidden layers, hidden units, and activation functions to fine-tune the model's performance.

4.2 Disadvantages

1. **Cold-start problem:** When dealing with new users or items with very few interactions or features, the model might face challenges in providing accurate recommendations since it hasn't learned enough about these entities.
2. **Computational complexity**: Training neural networks can be computationally intensive, especially if dealing with large embedding dimensions, numerous item features, or a vast number of users and items.

5. Training Process

- Epochs: 15
- Optimizer: Adam(lr=1e-3)
- Loss function: MSELoss (Mean Squared Error)
- Final Loss:
    - Train loss: 0.884
    - Val loss: 0.888
- Dropout: 0.3 (to prevent overfitting)
- Model Specification:
    - num_users: 944
    - num_movies: 1683
    - embed_dim: 64
    - hidden_dim: 64
    - item_featuers: 19 (genres)
- device: cpu

6. Evaluation

For evalutaion i used two metrics.
MSE: mean squared loss to check loss we will get on evaluation dataset.
Accuracy: Let’s think of predicting rating as a binary task if the rating is below 2.5 then I didn’t like the movie and if it’s above 2.5 I liked the movie. In that case I can use accuracy.
For example:

1. model prediction for a movie 3.56 (good movie), and my actual rating is 4 (good movie) then model predicted correctly.
2. Model prediction for a movie 2.3 (bad movie), and my rating is 2 (bad movie) then model predicted correctly.

3. Model prediction for a movie 2.4 (bad movie), and my rating is 5 (good movie) then model predicted incorrectly.

Using this metric I can measure how the model predictions is similar to mine.

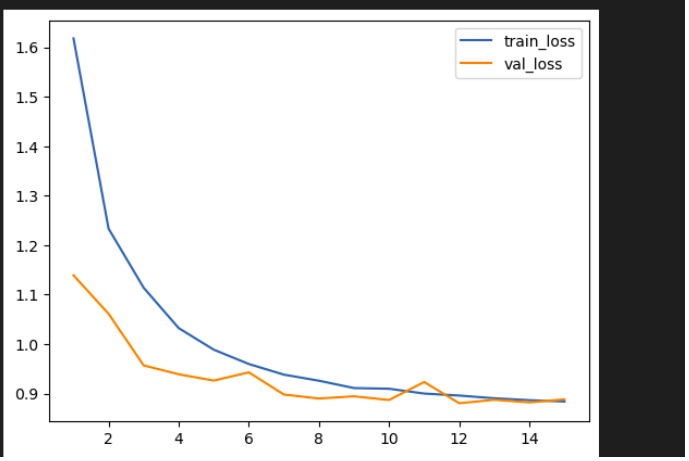
So my final mse and accuracy is:

- **MSE: 0.8272227096129285**
- **Accuracy: 0.85955**

7. Results

```
plt.plot(range(1, len(train_loss)+1), train_loss, label='train_loss')
plt.plot(range(1, len(val_loss)+1), val_loss, label='val_loss')
plt.legend()
plt.show()
```

0.8s



As we can see loss is descreasing gradually so the model is learning how to properly map user-item with rating. So it can predict correct rating given user id and some information about movie.

Example of the model prediction:

```
Already watched movies (top 10):
Kolya (1996)
Remains of the Day, The (1993)
Toy Story (1995)
Chasing Amy (1997)
Crumb (1994)
French Twist (Gazon maudit) (1995)
Fargo (1996)
Return of the Jedi (1983)
Dead Poets Society (1989)
Mr. Holland's Opus (1995)
==========================================

Predicted movies for user 1 (top 10):
Casablanca (1942)
Close Shave, A (1995)
Schindler's List (1993)
Pather Panchali (1955)
To Kill a Mockingbird (1962)
One Flew Over the Cuckoo's Nest (1975)
Manchurian Candidate, The (1962)
Prefontaine (1997)
Sunset Blvd. (1950)
North by Northwest (1959)
```