Recommendation Model

Overview

This is an implementation of the Neural Collaborative Filtering (NCF) framework with Neural Matrix Factorization (NeuMF) model as described in the Neural Collaborative Filtering paper. Current implementation is based on the code from the authors' NCF code and the Stanford implementation in the MLPerf Repo.

NCF is a general framework for collaborative filtering of recommendations in which a neural network architecture is used to model user-item interactions. Unlike traditional models, NCF does not resort to Matrix Factorization (MF) with an inner product on latent features of users and items. It replaces the inner product with a multi-layer perceptron that can learn an arbitrary function from data.

Two instantiations of NCF are Generalized Matrix Factorization (GMF) and Multi-Layer Perceptron (MLP). GMF applies a linear kernel to model the latent feature interactions, and MLP uses a nonlinear kernel to learn the interaction function from data. NeuMF is a fused model of GMF and MLP to better model the complex user-item interactions, and unifies the strengths of linearity of MF and non-linearity of MLP for modeling the user-item latent structures. NeuMF allows GMF and MLP to learn separate embeddings, and combines the two models by concatenating their last hidden layer. neumf_model.py defines the architecture details.

Some abbreviations used the code base include: - NCF: Neural Collaborative Filtering - NeuMF: Neural Matrix Factorization - GMF: Generalized Matrix Factorization - MLP: Multi-Layer Perceptron - HR: Hit Ratio (HR) - NDCG: Normalized Discounted Cumulative Gain - ml-1m: MovieLens 1 million dataset - ml-20m: MovieLens 20 million dataset

Dataset

The MovieLens datasets are used for model training and evaluation. Specifically, we use two datasets: **ml-1m** (short for MovieLens 1 million) and **ml-20m** (short for MovieLens 20 million).

ml-1m

ml-1m dataset contains 1,000,209 anonymous ratings of approximately 3,706 movies made by 6,040 users who joined MovieLens in 2000. All ratings are contained in the file "ratings.dat" without header row, and are in the following format:

```
UserID::MovieID::Rating::Timestamp
```

- UserIDs range between 1 and 6040.

- MovieIDs range between 1 and 3952.
- Ratings are made on a 5-star scale (whole-star ratings only).

ml-20m

ml-20m dataset contains 20,000,263 ratings of 26,744 movies by 138493 users. All ratings are contained in the file "ratings.csv". Each line of this file after the header row represents one rating of one movie by one user, and has the following format:

```
userId,movieId,rating,timestamp
```

- The lines within this file are ordered first by userId, then, within user, by movieId.
- Ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars).

In both datasets, the timestamp is represented in seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970. Each user has at least 20 ratings.

Running Code

Download and preprocess dataset

To download the dataset, please install Pandas package first. Then issue the following command:

```
python movielens.py
```

Arguments: * `--data_dir`: Directory where to download and save the preprocessed data. By default, it is `/tmp/movielens-data/`. * `--dataset`: The dataset name to be downloaded and preprocessed. By default, it is `ml-1m`.

Use the `--help` or `-h` flag to get a full list of possible arguments.

Note the ml-20m dataset is large (the rating file is ~500 MB), and it may take several minutes (~2 mins) for data preprocessing. Both the ml-1m and ml-20m datasets will be coerced into a common format when downloaded.

Train and evaluate model

ncf_keras_main.py is the Keras trainer that supports features in TF 2.x. Users can train the model on both GPU and TPU.

To train and evaluate the model, issue the following command:

```
python ncf_keras_main.py
```

Arguments: * `--model_dir`: Directory to save model training checkpoints. By default, it is `/tmp/ncf/`. * `--data_dir`: This should be set to the same directory given to the `data_download`'s `data_dir` argument. * `--dataset`: The dataset name to be downloaded and preprocessed. By default, it is `ml-1m`. * `--num_gpus`: The number of GPUs used for training/evaluation of the model. Use CPU if this flag is 0. By default, it is 1.

There are other arguments about models and training process. Refer to the Flags package documentation or use the `--helpfull` flag to get a full list of possible arguments with detailed descriptions.