# Neural Collaborative Filtering

A modified version of Yi Hong's implementation of NCF in Pytorch.

## Dataset

- Raw data
  - MovieLens 1M Dataset
- Processed data
  - ratings.csv
  - time.csv
  - time_weighted_rating_movielens1m.csv

## Files

- `time_weighted_movielens1m.ipynb` : generate time weighted ratings from raw data
- `preprocess.py` : preprocesss raw data to be used in model training
- `preprocess-time.py` : preprocesss time weighted data to be used in model training
- `data.py` : prepare train/test dataset
- `train.py` : entry point to train the models
- `utils.py` : some handy functions for model training
- `metrics.py` : evaluation metrics including accuracy rate of rating prediction
- `gmf.py` : generalized matrix factorization model
- `mlp.py` : multi-layer perceptron model
- `cnn.py` : convolutional neural network model
- `neumf.py` : ensemble of gmf, mlp and cnn
- `engine.py` : training engine
- `proxy.py` : calcluating custom metric used as a proxy of user retention rate

## Directiories

- `./src/checkpoints` : checkpoints of model state for each epoch during training
- `./src/data/predicted` : predicted ratings
- `./src/data/processed` : processed data for model training
- `./src/data/raw` : raw dataset
- `./src/epoch100` : final model states

## Dependencies

- Pytorch
- Numpy
- Pandas
- TensorboardX
- SciKit Learn
- Statsmodels

- Matplotlib

> [!IMPORTANT]
> From this point onwards, please ensure your current working directory is `./src`.
> All scripts should be run from the `./src` directory.

# Data Preprocessing

- Download the raw data from grouplens
- Create the folders
  - `./src/data/raw`
  - `./src/data/processed`
  - `./src/checkpoints`
- Extract the raw data to `./src/data/raw`
  - The raw data folder should contain the file `ratings.dat` as such: `./src/data/raw/ratings.dat`
- Run the `./src/time_weighted_movielens1m.ipynb` to get the time weighted ratings dataset
  - The notebook will create a csv file `./src/data/processed/time_weighted_rating_movielens1m.csv`
- Run `preprocess.py` to preprocess the data
  - The script will preprocess the raw `ratings.dat` file and store it in a csv file `./src/data/processed/ratings.csv`
- Run `preprocess-time.py` to preprocess the time weighted data
  - The script will preprocess the `time_weighted_rating_movielens1m.csv` file and store it in a csv file `./src/data/processed/time.csv`
- Your final file structure should look like this

```
- ./src
  - /data
    - /raw
      - ratings.dat
    - /processed
      - ratings.csv
      - time.csv
      - time_weighted_rating_movielens1m.csv
```

# Model Configuration

- All model configurations are kept in `./src/config.py` in the `get_configs` function which is used to initialise all model configs
- If CUDA is available or using Apple Scilicon, you can enable the gpu flags in the base config to speed up training

```
# ./src/config.py
def get_configs(num_user, num_item):
  base_config = {
    "use_cuda": False, # set to true if CUDA is available
```

```
    "use_mps": False, # set to true if using Apple Scilicon and Metal API is
available
    # ...
  }
  # ...
```

# Model Training

- Run `train.py` with the following commands to pretrain the individual models.
  Use the `data` flag to choose which dataset to train the model on.

```
# run separately to train models individually
python3 train.py --model="gmf" --data="[ratings.csv|time.csv]"
python3 train.py --model="mlp" --data="[ratings.csv|time.csv]"
python3 train.py --model="cnn" --data="[ratings.csv|time.csv]"
```

  - Checkpoints for the model state will be generated for each epoch
- Replace the filenames (from `./src/checkpoints`) in `neumf_config` to load the pretrained model weights.

```
# ./src/config.py
def get_configs(num_user, num_item):
  # ...
  neumf_config = {
    # ...
    "pretrain_mf": "<full path relative to ./src>",
    "pretrain_mlp": "<full path relative to ./src>",
    "pretrain_cnn": "<full path relative to ./src>",
    # ...
  }
  # ...
```

- Run `train.py` with the `neumf` flag to train the final ensemble NCF model

```
python3 train.py --model="neumf" --data="[ratings.csv|time.csv]"
```

# Model Prediction

- For convenience, the model states for the final model is provided in `./src/epoch100`.
  These model states will be used for the prediction of user accuracies.
- To predict the user accuracies, run the following commands
  - Normal ratings

```
python3 predict.py --model="neumf" --state="neumf_normal.model" --
data="ratings.csv"
```

  - Time weighted ratings

```
python3 predict.py --model="neumf" --state="neumf_time.model" --
data="time.csv"
```

- The scripts will output a csv file in the `./src/data` folder.
  Please rename the files before running each command to prevent them from being overridden.
  - `predict_normal.csv` : result of the user predictions on `ratings.csv` dataset
  - `predict_time.csv` : result of the user predictions on `time.csv` dataset

# Model Evaluation

- Run the proxy script to calculate the metrics for user retention

```
python3 proxy.py
```

# References

## Codebase

https://github.com/yihong-chen/neural-collaborative-filtering

## NCF paper

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. (2017). Neural Collaborative Filtering. WWW 2017. https://doi.org/10.1145/3038912.3052569

## ONCF paper

He, X., Du, X., Wang, X., Tian, F., Tang, J., & Chua, T. (2018). Outer Product-based Neural Collaborative Filtering. IJCAI 2018. https://doi.org/10.24963/ijcai.2018/308