Algorithm Porting Project - CODIGEM

Publication title: CODIGEM

Introduction and brief description

In [1], published at *KSEM 2022*, the authors propose a collaborative filtering algorithm called **CODIGEM** (Collaborative Diffusion Generative Model). The model uses a denoising diffusion probabilistic model (DDPM) to address two common limitations of classical collaborative filtering methods: weak collaborative signals and poor latent representations.

The paper claims this approach captures complex, non-linear patterns from implicit feedback data by introducing noise in a forward process and reconstructing signals in a reverse process, both parameterized via deep neural networks. CODIGEM is designed to work with top-N recommendation tasks and shows competitive performance across multiple datasets.

Introduction checklist:

- The model performs top-N item recommendations to a user: **Yes**
- The model relies on sequential patterns or recurrent neural networks: **No**
- The model relies on non-structured input data such as images or text: **No**

Datasets

Evaluation is performed on three datasets, preprocessed for implicit feedback recommendation:

- **MovieLens-1M (ML-1M)**: User-movie ratings ≥ 3 retained; only users/items with ≥ 10 interactions. Ratings are binarized. [1]
- **MovieLens-20M (ML-20M)**: Same preprocessing as ML-1M. [2]
- **Amazon Electronics (AE)**: Amazon product ratings with same thresholds. [3]

Dataset	Interactions	Items	Users	Density
ML-20M	9,990,682	20,108	136,677	0.0036
ML-1M	834,449	3,124	6,034	0.0443
AE	234,521	8,361	13,456	0.0021

Table 1: Dataset Statistics

[1] `https://grouplens.org/datasets/movielens/1m/`
[2] `https://grouplens.org/datasets/movielens/20m/`
[3] `http://jmcauley.ucsd.edu/data/amazon/`

Datasets checklist:

- At least one of the datasets is publicly available for download **Yes**
- The characteristics (size and density) of the datasets you are using for the experiments (i.e., not those in the dataset description online, but those loaded by the DataReader considering any preprocessing that may be required) are consistent with the ones reported in the original article: **Yes**

Evaluation protocol

The dataset is split into 80% training, 10% validation, 10% test, per user. Evaluation compares each positive item to all unobserved items. Metrics reported:

- Recall@20, Recall@50
- NDCG@20, NDCG@50

Evaluation protocol checklist:

- The original train-test split of the data is available on the Github repository? **Yes**
- The original negative item split used for the evaluation phase (not for training) is available on the Github repository? **Not used**
- Does the paper mention a validation set? **Yes**
- Does the paper state that the evaluation is performed in the same way (data, metrics) as another article? **No**

Baseline algorithms checklist:

- Is TopPopular among the baselines? **Yes**
- Are ItemKNN or UserKNN with cosine similarity among the baselines? **Yes**
- Does the paper state if they include a shrink term? **No**

Hyperparameter tuning

The hyperparameters are tuned using Recall@20,Recall@50 ,NDCG@20, NDCG@50. The values of the hyperparameters for the proposed method are described in Table:

Hyperparameter	Described in	Value
β_t	source code	0.0001
T (diffusion steps)	source code	3
Learning rate	source code	0.001
Epochs	source code	100
Batch size	source code	200

Table 2: Hyperparameter values

Hyperparameter tuning checklist:

- Does the paper mention the metric-cutoff optimised during hyperparameter tuning? **Yes**
- Does the paper report the hyperparameter value range used during the tuning for the proposed method? **Yes**
- Does the paper report the hyperparameter value range used during the tuning for the baseline methods? **No**
- Does the paper state that the hyperparameters of a baseline are the default ones or are taken from another article? **No**
- Does the paper mention how the optimal number of epochs is selected? **Yes**
- Does the paper state that the value of one of the hyperparameters is fixed for all or some of the baselines? **Yes** (e.g. $T = 3$)

Source code

The source code is publicly available: `https://github.com/WorldChanger01/CODIGEM`

Dataset	**Source**	**R@20**	**R@50**	**N@20**	**N@50**
ML-1M	Paper	0.2796	0.4354	0.2447	0.3026
	Original source	-	-	-	-
	Ported source	0.0062542	0.0124521	0.0066509	0.0082983
ML-20M	Paper	0.3512	0.4698	0.3031	0.3414
	Original source	0.3386	0.4554	0.2953	0.3329
	Ported source	0.0005449	0.0038252	0.0007464	0.0019113
AE	Paper	0.0771	0.1338	0.0360	0.0478
	Original source	-	-	-	-
	Ported source	0.0001193	0.0003867	0.0000454	0.0001037

Table 3: Evaluation results for each dataset and implementation version

Source code checklist:

- Is the original source code executable with at most minor changes (e.g., fixing imports or downloading data)?**Yes**
- Is the original source code, with the correct hyperparameters, able to reproduce the results reported in the paper? **Yes. Example: ML-20M Recall@20 96%**
- Does the ported version of the original algorithm pass the test suite provided in Test/run unittest recommenders.py? **No**
- Is the ported version of the original algorithm, with the correct hyperparameters, able to reproduce the results obtained by the original source code? **No. Example: ML-1M Recall@20 2%**
- Is the source code delivered compliant with the requirements in Section ? **Yes**
- In the original source code is the test data used during the training in any way (for example to select the number of epochs or to select the negative samples...)? **No**

References

[1] Joojo Walker, Ting Zhong, Fengli Zhang, Qiang Gao, and Fan Zhou. Recommendation via collaborative diffusion generative model. In *KSEM*, volume 13370 of *Lecture Notes in Artificial Intelligence*, pages 593–605. Springer, 2022.