reco Documentation

Release 0.1.1

Mayukh Bhattacharyya

December 01, 2016

Contents

1 Installation 3

2 Tutorials 5

3 Modules 7

reco is a recommendation systems library in python. It currently has SVD and Collaborative Filtering available. Explore and have fun.

CHAPTER 1

Installation

Installing reco through pip is easy. Use a terminal or command prompt and type out

```
pip install reco
```

After this you can import reco in your scripts.

Tutorials

2.1 SVD Recommender Tutorial

Below is a tutorial on using the SVD recommender module.:

```
from reco.recommender import SVDRecommender
from reco.datasets import load_movielens

data = load_movielens()


svd = SVDRecommender(no_of_features=4)

# Creates the user-item matrix, the userIds on the rows and the itemIds on the columns.
user_item_matrix, users, items = svd.create_utility_matrix(data, formatizer={'user':'userId', 'item'

# fits the svd model to the matrix data.
svd.fit(user_item_matrix, users, items)

##### TESTING #####

test_users = [1, 65, 444, 321]

# recommends 4 undiscovered items per each user
results = svd.recommend(test_users, N=4)

# outputs 5 most similar users to user with userId 65
similars = svd.topN_similar(x=65, N=5, column='user')

print(results)
print(similars)

##### The End #####
```

Modules

3.1 Recommenders

There are 2 recommenders available

- SVDRecommender
- CFRecommender

3.1.1 CFRecommender

This is the collaborative filtering module. It has 3 similarity coefficients available to choose from, namely pearson, jaccard and cosine.

```
reco.recommender.CFRecommender(formatizer = {'user':0,'item':1,'value':2}, sim_engine = 'pearson')
```

Parameters

formatizer :

The format of the datasets that the recommender will be working upon. The values corresponding to every key is the column number for that key. For example the default format is that user_ids are in first column, then item_ids and then the corresponding ratings or values.

sim_engine :

The similarity coefficient for measuring the similarity between entities (users or items). The available options are ‘pearson’,’jaccard’,’cosine’.

Methods

fit:

Fits the data to the recommender.

```
fit(data, formatizer)
```

Parameters

- data: The dataset.
- formatizer: Overrides the format during initialization for this fit method.

predict:

Predicts the rating or value for each user-item pair in the rows of the X.

```
predict(X, formatizer=None)
```

Parameters

- X: The dataset for prediction.
- formatizer: Overrides the format during initialization for this predict method.

getRecommendations:

Recommends top N items for the user given (reducing N does not improve speed):

```
getRecommendations(user, score=True, N=10)
```

Parameters

- user: The user_id for which to recommend.
- score: Whether or not to give out the predicted rating for each item. If score is True, output is a list of tuples.
- N: Number of items to recommend.

topMatches:

Gives the top N similar users for the user given (reducing N does not improve speed):

```
topMatches(self, user, score = True, N=10)
```

Parameters

- user: The user_id for which to process.
- score: Whether or not to give out the similarity coefficient for each user. If score is True, output is a list of tuples.
- N: Number of users to find.

3.1.2 SVDRecommender

This is the singular vector decomposition module. It breaks the dataset as feature vectors of each user and each item.

```
reco.recommender.SVDRecommender(no_of_features = 15, method = 'default')
```

Parameters

no_of_features :

The number of features in the feature vectors of each user and each item that the dataset is decomposed into.

method :

'default' is the only option at this moment.

Methods

create_utility_matrix:

Creates the user-item matrix from the dataset. The user-item matrix and the users list and items list are utilized in the fit method.

```
create_utility_matrix(self, data, formatizer = {'user':0, 'item': 1, 'value': 2})
```

Parameters

- data: The dataset.
- formatizer: Stores the column names/ids for the users, items and ratings columns as a dictionary.

fit:

Fits the data to the recommender.

```
fit(user_item_matrix, userList, itemList)
```

Parameters

- user_item_matrix: The data represented as an user-item matrix. The rows represent the users and the columns represent the items.
- userList: The users or names/ids of the row elements in the correct order as to the the user_item_matrix.
- itemList: The items or names/ids of the column elements in the correct order as to the the user_item_matrix.

recommend:

Gives out a recommended ranked list of undiscovered items for each user given in a list. Will not recommend an item which the user has already rated.:

```
recommend(users_list, N=10, values = False)
```

Parameters

- users_list: The users in a list for each of which the items are to be recommended.
- N: Number of items to be recommended. Recommends only the undiscovered items, i.e. items for which the user had no rating in the user-item matrix. Default value is 10.
- values: Whether the predicted rating for the item is to be given as output. If set to True, output for each user will be a list of tuples (item, predicted_rating).

predict:

Predicts the rating or value for each user-item pair in the rows of the X as a list.

```
predict(X, formatizer = {'user':0, 'item': 1, 'value': 2})
```

Parameters

- X: The dataset having the user and item on each rows whose corresponding rating is to be predicted.
- formatizer: Stores the column names/ids for the users and items columns as a dictionary.

topN_similar:

Predicts the rating or value for each user-item pair in the rows of the X as a list.

```
topN_similar(x, column='item', N=10, weight=True)
```

Parameters

- x: The id for the user or item.
- column: 'item' if x is an item or 'user' if x is an user.
- N: Number of similar entities to find.
- weight: Give the associateds weights of similarity.

3.2 Metrics

reco has some metrics that you can use to assess the performance of your recommenders.

3.2.1 rmse:

Root Mean Square Error. It is useful in assessing the performance of the predict methods, i.e. in assessing the accuracy of the predicted ratings.

```
reco.metrics.rmse(true, predicted)
```

- true: True value list.

*predicted: predicted value list.

3.2.2 kendalltau:

Measures the Kendal-Tau correlation between 2 ranked lists. Useful in assessing the performance of the recommend methods, i.e. the accuracy of the orders in which the items are recommended. For example, better or more relevant items should be at the start.

```
reco.metrics.kendalltau(rankA, rankB)
```

rankA and rankB must be of the same length. Kendal Tau measures the similarity in the order of elements in the 2 ranked lists.

3.3 Datasets

reco has some datasets in store. Here's how to load them and start using them.

3.3.1 Movielens100k Dataset

Loads the famous Movielens100k dataset as a pandas dataframe.

```
reco.datasets.load_movielens()
```