Cornac Documentation

Release 0.1.0.post5

Cornac Contributors

Mar 31, 2019

Getting Started

Cornac is python recommender systems library for **easy**, **effective** and **efficient** experiments. Cornac is **simple** and **handy**. It is designed from the ground-up to faithfully reflect the standard steps taken by researchers to implement and evaluate personalized recommendation models. Moreover, contributing new recommender models, evaluation metrics, etc., to Cornac is very easy and smooth. For instance, if you already have a python implementation of your model, e.g., PMF, you will need to spend less than 5 minutes in average to integrate it to Cornac.

CHAPTER 1

Installation

Currently, we are supporting Python 3 (version 3.6 is recommended). There are several ways to install Cornac:

- **From PyPI (you may need a C++ compiler)**:

```
pip3 install cornac
```

- **From Anaconda**:

```
conda install cornac -c qttruong -c pytorch
```

- **From the GitHub source (for latest updates)**:

```
pip3 install Cython
git clone https://github.com/PreferredAI/cornac.git
cd cornac
python3 setup.py install
```

Note:

Additional dependencies required by models are listed here.

Some of the algorithms use *OpenMP* to support multi-threading. For OSX users, in order to run those algorithms efficiently, you might need to install *gcc* from Homebrew to have an OpenMP compiler:

```
brew install gcc | brew link gcc
```

If you want to utilize your GPUs, you might consider:

- TensorFlow installation instructions: https://www.tensorflow.org/install/
- PyTorch installation instructions: https://pytorch.org/get-started/locally/
- cuDNN: https://docs.nvidia.com/deeplearning/sdk/cudnn-install/ (for Nvidia GPUs)

CHAPTER 2

First example

This example will show you how to run your very first experiment using Cornac.

```
import cornac as cn

# Load MovieLens 100K dataset
ml_100k = cn.datasets.movielens.load_100k()

# Split data based on ratio
ratio_split = cn.eval_methods.RatioSplit(data=ml_100k,
                                         test_size=0.2,
                                         rating_threshold=4.0,
                                         seed=123)

# Here we are comparing: Biased MF, PMF, and BPR
mf = cn.models.MF(k=10, max_iter=25, learning_rate=0.01, lambda_reg=0.02, use_
↪bias=True)
pmf = cn.models.PMF(k=10, max_iter=100, learning_rate=0.001, lamda=0.001)
bpr = cn.models.BPR(k=10, max_iter=200, learning_rate=0.01, lambda_reg=0.01)

# Define metrics used to evaluate the models
mae = cn.metrics.MAE()
rmse = cn.metrics.RMSE()
rec_20 = cn.metrics.Recall(k=20)
ndcg_20 = cn.metrics.NDCG(k=20)
auc = cn.metrics.AUC()

# Put it together into an experiment and run
exp = cn.Experiment(eval_method=ratio_split,
                    models=[mf, pmf, bpr],
                    metrics=[mae, rmse, rec_20, ndcg_20, auc],
                    user_based=True)
exp.run()
```

Output:

	MAE	RMSE	Recall@20	NDCG@20	AUC	Train (s)	Test (s)
MF	0.7441	0.9007	0.0622	0.0534	0.2952	0.0736	8.1187
PMF	0.7493	0.9084	0.0835	0.0673	0.4749	358.7642	8.4184
BPR	1.5595	1.8864	0.0744	0.0657	0.5932	2.5395	8.5734

CHAPTER 3

Data

class cornac.data.**FeatureModule**(*features=None*, *ids=None*, *normalized=False*, ***kwargs*)

Parameters

- **features** (*numpy.ndarray or scipy.sparse.csr_matrix, default = None*) – Numpy 2d-array that the row indices are aligned with user/item in *ids*.
- **ids** (*List, default = None*) – List of user/item ids that the indices are aligned with *corpus*. If None, the indices of provided *features* will be used as *ids*.

batch_feature(*batch_ids*)
Return a matrix (batch of feature vectors) corresponding to provided batch_ids

build(*id_map=None*)
Build the feature matrix. Features will be swapped if the id_map is provided

feature_dim
Return the dimensionality of the feature vectors

features
Return the whole feature matrix

class cornac.data.**TextModule**(*corpus: List[str] = None*, *ids: List = None*, *tokenizer: cornac.data.text.Tokenizer = None*, *vocab: cornac.data.text.Vocabulary = None*, *max_vocab: int = None*, *max_doc_freq: Union[float, int] = 1.0*, *min_freq: int = 1*, *stop_words: Union[List, str] = None*, ***kwargs*)

Text module

Parameters

- **corpus** (*List[str], default = None*) – List of user/item texts that the indices are aligned with *ids*.
- **ids** (*List, default = None*) – List of user/item ids that the indices are aligned with *corpus*. If None, the indices of provided *corpus* will be used as *ids*.

- **tokenizer** (*Tokenizer, optional, default = None*) – Tokenizer for text splitting. If None, the BaseTokenizer will be used.
- **vocab** (*Vocabulary, optional, default = None*) – Vocabulary of tokens. It contains mapping between tokens to their integer ids and vice versa.
- **max_vocab** (*int, optional, default = None*) – The maximum size of the vocabulary. If vocab is provided, this will be ignored.
- **max_doc_freq** (*Union[float, int] = 1.0*) – The maximum frequency of tokens appearing in documents to be excluded from vocabulary. If float, the value represents a proportion of documents, int for absolute counts. If *vocab* is not None, this will be ignored.
- **min_freq** (*int, default = 1*) – The minimum frequency of tokens to be included into vocabulary. If *vocab* is not None, this will be ignored.
- **stop_words** (*Collection, str, default: None*) – Collection of stop words which will be ignored when building *Vocabulary*. If str, it indicates a built-in stop words list. Currently, only *english* is supported.

batch_seq(*batch_ids*, *max_length=None*)
Return a numpy matrix of text sequences containing token ids with size=(len(batch_ids), max_length). If max_length=None, it will be inferred based on retrieved sequences.

batch_tfidf(*batch_ids*)
Return matrix of TF-IDF features corresponding to provided batch_ids

build(*id_map=None*)
Build the model based on provided list of ordered ids

class cornac.data.**ImageModule**(***kwargs*)
Image module

batch_image(*batch_ids*, *target_size=(256*, *256)*, *color_mode='rgb'*, *interpolation='nearest'*)
Return batch of images corresponding to provided batch_ids

build(*id_map=None*)
Build the model based on provided list of ordered ids

class cornac.data.**GraphModule**(***kwargs*)
Graph module

batch(*batch_ids*)
Return batch of vectors from the sparse adjacency matrix corresponding to provided batch_ids.

Parameters **batch_ids** (*array, required*) – An array contains the ids of rows to be returned from the sparse adjacency matrix.

build(*id_map=None*)
Build the feature matrix. Features will be swapped if the id_map is provided

get_train_triplet(*train_row_ids*, *train_col_ids*)
Get the training tuples

class cornac.data.**TrainSet**(*uid_map*, *iid_map*)
Training Set

Parameters

- **uid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of users.
- **iid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of items.

get_iid(*raw_iid*)
Return the mapped id of an item given a raw id

get_uid(*raw_uid*)
Return the mapped id of a user given a raw id

static idx_iter(*idx_range*, *batch_size=1*, *shuffle=False*)
Create an iterator over batch of indices

Parameters

- **batch_size** (*int, optional, default = 1*) –
- **shuffle** (*bool, optional*) – If True, orders of triplets will be randomized. If False, default orders kept

Returns iterator

Return type batch of indices (array of np.int)

iid_list
Return the list of mapped item ids

is_unk_item(*mapped_iid*)
Return whether or not an item is unknown given the mapped id

is_unk_user(*mapped_uid*)
Return whether or not a user is unknown given the mapped id

num_items
Return the number of items

num_users
Return the number of users

raw_iid_list
Return the list of raw item ids

raw_uid_list
Return the list of raw user ids

uid_list
Return the list of mapped user ids

class cornac.data.**MatrixTrainSet**(*matrix*, *max_rating*, *min_rating*, *global_mean*, *uid_map*, *iid_map*)
Training set contains preference matrix

Parameters

- **matrix** (scipy.sparse.csr_matrix) – Preferences in the form of scipy sparse matrix.
- **max_rating** (*float*) – Maximum value of the preferences.
- **min_rating** (*float*) – Minimum value of the preferences.
- **global_mean** (*float*) – Average value of the preferences.
- **uid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of users.
- **iid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of items.

classmethod from_uir(*data*, *global_uid_map=None*, *global_iid_map=None*, *global_ui_set=None*, *verbose=False*)

Constructing TrainSet from triplet data.

Parameters

- **data** (*array-like, shape: [n_examples, 3]*) – Data in the form of triplets (user, item, rating)
- **global_uid_map** (defaultdict, optional, default: None) – The dictionary containing global mapping from original ids to mapped ids of users.
- **global_iid_map** (defaultdict, optional, default: None) – The dictionary containing global mapping from original ids to mapped ids of items.
- **global_ui_set** (set, optional, default: None) – The global set of tuples (user, item). This helps avoiding duplicate observations.
- **verbose** (*bool, default: False*) – The verbosity flag.

Returns **train_set** – MatrixTrainSet object.

Return type <cornac.data.MatrixTrainSet>

item_iter(*batch_size=1*, *shuffle=False*)

Create an iterator over item ids

Parameters

- **batch_size** (*int, optional, default = 1*) –
- **shuffle** (*bool, optional*) – If True, orders of triplets will be randomized. If False, default orders kept

Returns **iterator**

Return type batch of item ids (array of np.int)

uij_iter(*batch_size=1*, *shuffle=False*)

Create an iterator over data yielding batch of users, positive items, and negative items

Parameters

- **batch_size** (*int, optional, default = 1*) –
- **shuffle** (*bool, optional*) – If True, orders of triplets will be randomized. If False, default orders kept

Returns **iterator** – batch of negative items (array of np.int)

Return type batch of users (array of np.int), batch of positive items (array of np.int),

uir_iter(*batch_size=1*, *shuffle=False*)

Create an iterator over data yielding batch of users, items, and rating values

Parameters

- **batch_size** (*int, optional, default = 1*) –
- **shuffle** (*bool, optional*) – If True, orders of triplets will be randomized. If False, default orders kept

Returns **iterator** – batch of ratings (array of np.float)

Return type batch of users (array of np.int), batch of items (array of np.int),

user_iter(*batch_size=1*, *shuffle=False*)

Create an iterator over user ids

Parameters

- **batch_size** (*int, optional, default = 1*) –
- **shuffle** (*bool, optional*) – If True, orders of triplets will be randomized. If False, default orders kept

Returns iterator

Return type batch of user ids (array of np.int)

class cornac.data.**MultimodalTrainSet**(*matrix*, *max_rating*, *min_rating*, *global_mean*, *uid_map*, *iid_map*, ***kwargs*)

Multimodal training set

Parameters

- **matrix** (scipy.sparse.csr_matrix) – Preferences in the form of scipy sparse matrix.
- **max_rating** (*float*) – Maximum value of the preferences.
- **min_rating** (*float*) – Minimum value of the preferences.
- **global_mean** (*float*) – Average value of the preferences.
- **uid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of users.
- **iid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of items.

class cornac.data.**TestSet**(*user_ratings*, *uid_map*, *iid_map*)

Test Set

Parameters

- **user_ratings** (defaultdict of list) – The dictionary containing lists of tuples of the form (item, rating). The keys are user ids.
- **uid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of users.
- **iid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of items.

classmethod from_uir(*data*, *global_uid_map*, *global_iid_map*, *global_ui_set*, *verbose=False*)

Constructing TestSet from triplet data.

Parameters

- **data** (*array-like, shape: [n_examples, 3]*) – Data in the form of triplets (user, item, rating)
- **global_uid_map** (defaultdict) – The dictionary containing global mapping from original ids to mapped ids of users.
- **global_iid_map** (defaultdict) – The dictionary containing global mapping from original ids to mapped ids of items.
- **global_ui_set** (set) – The global set of tuples (user, item). This helps avoiding duplicate observations.
- **verbose** (*bool, default: False*) – The verbosity flag.

Returns **test_set** – TestSet object.

Return type <cornac.data.TestSet>

get_iid(*raw_iid*)
Return the mapped id of an item given a raw id

get_ratings(*mapped_uid*)
Return a list of tuples of (item, rating) of given mapped user id

get_uid(*raw_uid*)
Return the mapped id of a user given a raw id

users
Return a list of users

class cornac.data.**MultimodalTestSet**(*user_ratings*, *uid_map*, *iid_map*, ***kwargs*)
Test Set

Parameters

- **user_ratings** (defaultdict of list) – The dictionary containing lists of tuples of the form (item, rating). The keys are user ids.
- **uid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of users.
- **iid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of items.

3.1 Train Set

@author: Quoc-Tuan Truong <tuantq.vnu@gmail.com>

class cornac.data.trainset.**MatrixTrainSet**(*matrix*, *max_rating*, *min_rating*, *global_mean*, *uid_map*, *iid_map*)
Training set contains preference matrix

Parameters

- **matrix** (scipy.sparse.csr_matrix) – Preferences in the form of scipy sparse matrix.
- **max_rating** (*float*) – Maximum value of the preferences.
- **min_rating** (*float*) – Minimum value of the preferences.
- **global_mean** (*float*) – Average value of the preferences.
- **uid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of users.
- **iid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of items.

classmethod from_uir(*data*, *global_uid_map=None*, *global_iid_map=None*, *global_ui_set=None*, *verbose=False*)
Constructing TrainSet from triplet data.

Parameters

- **data** (*array-like, shape: [n_examples, 3]*) – Data in the form of triplets (user, item, rating)

- **global_uid_map** (defaultdict, optional, default: None) – The dictionary containing global mapping from original ids to mapped ids of users.
- **global_iid_map** (defaultdict, optional, default: None) – The dictionary containing global mapping from original ids to mapped ids of items.
- **global_ui_set** (set, optional, default: None) – The global set of tuples (user, item). This helps avoiding duplicate observations.
- **verbose** (*bool, default: False*) – The verbosity flag.

Returns **train_set** – MatrixTrainSet object.

Return type <cornac.data.MatrixTrainSet>

item_iter(*batch_size=1*, *shuffle=False*)
Create an iterator over item ids

Parameters

- **batch_size** (*int, optional, default = 1*) –
- **shuffle** (*bool, optional*) – If True, orders of triplets will be randomized. If False, default orders kept

Returns **iterator**

Return type batch of item ids (array of np.int)

uij_iter(*batch_size=1*, *shuffle=False*)
Create an iterator over data yielding batch of users, positive items, and negative items

Parameters

- **batch_size** (*int, optional, default = 1*) –
- **shuffle** (*bool, optional*) – If True, orders of triplets will be randomized. If False, default orders kept

Returns **iterator** – batch of negative items (array of np.int)

Return type batch of users (array of np.int), batch of positive items (array of np.int),

uir_iter(*batch_size=1*, *shuffle=False*)
Create an iterator over data yielding batch of users, items, and rating values

Parameters

- **batch_size** (*int, optional, default = 1*) –
- **shuffle** (*bool, optional*) – If True, orders of triplets will be randomized. If False, default orders kept

Returns **iterator** – batch of ratings (array of np.float)

Return type batch of users (array of np.int), batch of items (array of np.int),

user_iter(*batch_size=1*, *shuffle=False*)
Create an iterator over user ids

Parameters

- **batch_size** (*int, optional, default = 1*) –
- **shuffle** (*bool, optional*) – If True, orders of triplets will be randomized. If False, default orders kept

Returns **iterator**

Return type batch of user ids (array of np.int)

class cornac.data.trainset.**MultimodalTrainSet**(*matrix*, *max_rating*, *min_rating*, *global_mean*, *uid_map*, *iid_map*, ***kwargs*)

Multimodal training set

Parameters

- **matrix** (scipy.sparse.csr_matrix) – Preferences in the form of scipy sparse matrix.
- **max_rating** (*float*) – Maximum value of the preferences.
- **min_rating** (*float*) – Minimum value of the preferences.
- **global_mean** (*float*) – Average value of the preferences.
- **uid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of users.
- **iid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of items.

class cornac.data.trainset.**TrainSet**(*uid_map*, *iid_map*)

Training Set

Parameters

- **uid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of users.
- **iid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of items.

get_iid(*raw_iid*)

Return the mapped id of an item given a raw id

get_uid(*raw_uid*)

Return the mapped id of a user given a raw id

static idx_iter(*idx_range*, *batch_size=1*, *shuffle=False*)

Create an iterator over batch of indices

Parameters

- **batch_size** (*int, optional, default = 1*) –
- **shuffle** (*bool, optional*) – If True, orders of triplets will be randomized. If False, default orders kept

Returns iterator

Return type batch of indices (array of np.int)

iid_list

Return the list of mapped item ids

is_unk_item(*mapped_iid*)

Return whether or not an item is unknown given the mapped id

is_unk_user(*mapped_uid*)

Return whether or not a user is unknown given the mapped id

num_items

Return the number of items

num_users
Return the number of users

raw_iid_list
Return the list of raw item ids

raw_uid_list
Return the list of raw user ids

uid_list
Return the list of mapped user ids

3.2 Test Set

@author: Quoc-Tuan Truong <tuantq.vnu@gmail.com>

class cornac.data.testset.**MultimodalTestSet**(*user_ratings*, *uid_map*, *iid_map*, ***kwargs*)
Test Set

Parameters

- **user_ratings** (defaultdict of list) – The dictionary containing lists of tuples of the form (item, rating). The keys are user ids.
- **uid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of users.
- **iid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of items.

class cornac.data.testset.**TestSet**(*user_ratings*, *uid_map*, *iid_map*)
Test Set

Parameters

- **user_ratings** (defaultdict of list) – The dictionary containing lists of tuples of the form (item, rating). The keys are user ids.
- **uid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of users.
- **iid_map** (defaultdict) – The dictionary containing mapping from original ids to mapped ids of items.

classmethod from_uir(*data*, *global_uid_map*, *global_iid_map*, *global_ui_set*, *verbose=False*)
Constructing TestSet from triplet data.

Parameters

- **data** (*array-like, shape: [n_examples, 3]*) – Data in the form of triplets (user, item, rating)
- **global_uid_map** (defaultdict) – The dictionary containing global mapping from original ids to mapped ids of users.
- **global_iid_map** (defaultdict) – The dictionary containing global mapping from original ids to mapped ids of items.
- **global_ui_set** (set) – The global set of tuples (user, item). This helps avoiding duplicate observations.
- **verbose** (*bool, default: False*) – The verbosity flag.

Returns **test_set** – TestSet object.

Return type `<cornac.data.TestSet>`

get_iid(*raw_iid*)
Return the mapped id of an item given a raw id

get_ratings(*mapped_uid*)
Return a list of tuples of (item, rating) of given mapped user id

get_uid(*raw_uid*)
Return the mapped id of a user given a raw id

users
Return a list of users

3.3 Graph Module

@author: Aghiles Salah <asalah@smu.edu.sg>

class cornac.data.graph.**GraphModule**(***kwargs*)
Graph module

batch(*batch_ids*)
Return batch of vectors from the sparse adjacency matrix corresponding to provided batch_ids.

Parameters **batch_ids** (*array, required*) – An array contains the ids of rows to be returned from the sparse adjacency matrix.

build(*id_map=None*)
Build the feature matrix. Features will be swapped if the id_map is provided

get_train_triplet(*train_row_ids*, *train_col_ids*)
Get the training tuples

3.4 Text Module

@author: Quoc-Tuan Truong <tuantq.vnu@gmail.com>

class cornac.data.text.**Tokenizer**
Generic class for other subclasses to extend from. This typically either splits text into word tokens or character tokens.

batch_tokenize(*texts: List[str]*) → List[List[str]]
Splitting a corpus with multiple text documents.

Returns **tokens**

Return type `List[List[str]]`

tokenize(*t: str*) → List[str]
Splitting text into tokens.

Returns **tokens**

Return type `List[str]`

class cornac.data.text.**BaseTokenizer**(*sep: str = ' '*, *pre_rules: List[Callable[str, str]] = None*, *stop_words: Union[List, str] = None*)
A base tokenizer use a provided delimiter *sep* to split text.

batch_tokenize(*texts: List[str]*) → List[List[str]]
Splitting a corpus with multiple text documents.

Returns tokens

Return type List[List[str]]

tokenize(*t: str*) → List[str]
Splitting text into tokens.

Returns tokens

Return type List[str]

class cornac.data.text.**Vocabulary**(*idx2tok: List[str], use_special_tokens: bool = False*)
Vocabulary basically contains mapping between numbers and tokens and vice versa.

classmethod from_sequences(*sequences: List[List[str]], max_vocab: int = None, min_freq: int = 1, use_special_tokens: bool = False*) → cornac.data.text.Vocabulary
Build a vocabulary from sequences (list of list of tokens).

classmethod from_tokens(*tokens: List[str], max_vocab: int = None, min_freq: int = 1, use_special_tokens: bool = False*) → cornac.data.text.Vocabulary
Build a vocabulary from list of tokens.

classmethod load(*path*)
Load a vocabulary from *path* to a pickle file.

save(*path*)
Save idx2tok into a pickle file.

to_idx(*tokens: List[str]*) → List[int]
Convert a list of *tokens* to their integer indices.

to_text(*indices: List[int], sep=' '*) → List[str]
Convert a list of integer *indices* to their tokens.

class cornac.data.text.**CountVectorizer**(*tokenizer: cornac.data.text.Tokenizer = None, vocab: cornac.data.text.Vocabulary = None, max_doc_freq: Union[float, int] = 1.0, min_freq: int = 1, max_features: int = None, stop_words: Union[List, str] = None, binary: bool = False*)
Convert a collection of text documents to a matrix of token counts This implementation produces a sparse representation of the counts using scipy.sparse.csr_matrix.

Parameters

- **tokenizer** (Tokenizer, *optional, default = None*) – Tokenizer for text splitting. If None, the BaseTokenizer will be used.
- **vocab** (Vocabulary, *optional, default = None*) – Vocabulary of tokens. It contains mapping between tokens to their integer ids and vice versa.
- **max_doc_freq** (*Union[float, int] = 1.0*) – The maximum frequency of tokens appearing in documents to be excluded from vocabulary. If float, the value represents a proportion of documents, int for absolute counts. If *vocab* is not None, this will be ignored.
- **min_freq** (*int, default = 1*) – The minimum frequency of tokens to be included into vocabulary. If *vocab* is not None, this will be ignored.
- **max_features** (*int, default=None*) – If not None, build a vocabulary that only consider the top *max_features* ordered by term frequency across the corpus. If *vocab* is not None, this will be ignored.

- **stop_words** (*Collection, str, default: None*) – Collection of stop words which will be ignored when building *Vocabulary*. If str, it indicates a built-in stop words list. Currently, only *english* is supported.
- **binary** (*boolean, default=False*) – If True, all non zero counts are set to 1.

fit (*raw_documents: List[str]*) → cornac.data.text.CountVectorizer

Build a vocabulary of all tokens in the raw documents.

Parameters **raw_documents** (*iterable*) – An iterable which yields either str, unicode or file objects.

Returns

Return type self

fit_transform (*raw_documents: List[str]) -> (typing.List[typing.List[str]], <class 'scipy.sparse.csr.csr_matrix'>*)

Build the vocabulary and return term-document matrix.

Parameters **raw_documents** (*List[str]*) –

Returns

sequences: List[List[str] Tokenized sequences of raw_documents

X: array, [n_samples, n_features] Document-term matrix.

Return type (sequences, X)

transform (*raw_documents: List[str]) -> (typing.List[typing.List[str]], <class 'scipy.sparse.csr.csr_matrix'>*)

Transform documents to document-term matrix.

Parameters **raw_documents** (*List[str]*) –

Returns

sequences: List[List[str] Tokenized sequences of raw_documents.

X: array, [n_samples, n_features] Document-term matrix.

Return type (sequences, X)

class cornac.data.text.**TextModule** (*corpus: List[str] = None, ids: List = None, tokenizer: cornac.data.text.Tokenizer = None, vocab: cornac.data.text.Vocabulary = None, max_vocab: int = None, max_doc_freq: Union[float, int] = 1.0, min_freq: int = 1, stop_words: Union[List, str] = None, **kwargs*)

Text module

Parameters

- **corpus** (*List[str], default = None*) – List of user/item texts that the indices are aligned with *ids*.
- **ids** (*List, default = None*) – List of user/item ids that the indices are aligned with *corpus*. If None, the indices of provided *corpus* will be used as *ids*.
- **tokenizer** (*Tokenizer, optional, default = None*) – Tokenizer for text splitting. If None, the BaseTokenizer will be used.
- **vocab** (*Vocabulary, optional, default = None*) – Vocabulary of tokens. It contains mapping between tokens to their integer ids and vice versa.

- **max_vocab** (*int, optional, default = None*) – The maximum size of the vocabulary. If vocab is provided, this will be ignored.
- **max_doc_freq** (*Union[float, int] = 1.0*) – The maximum frequency of tokens appearing in documents to be excluded from vocabulary. If float, the value represents a proportion of documents, int for absolute counts. If *vocab* is not None, this will be ignored.
- **min_freq** (*int, default = 1*) – The minimum frequency of tokens to be included into vocabulary. If *vocab* is not None, this will be ignored.
- **stop_words** (*Collection, str, default: None*) – Collection of stop words which will be ignored when building *Vocabulary*. If str, it indicates a built-in stop words list. Currently, only *english* is supported.

batch_seq(*batch_ids*, *max_length=None*)
Return a numpy matrix of text sequences containing token ids with size=(len(batch_ids), max_length). If max_length=None, it will be inferred based on retrieved sequences.

batch_tfidf(*batch_ids*)
Return matrix of TF-IDF features corresponding to provided batch_ids

build(*id_map=None*)
Build the model based on provided list of ordered ids

3.5 Image Module

@author: Quoc-Tuan Truong <tuantq.vnu@gmail.com>

class cornac.data.image.**ImageModule**(***kwargs*)
Image module

batch_image(*batch_ids*, *target_size=(256, 256)*, *color_mode='rgb'*, *interpolation='nearest'*)
Return batch of images corresponding to provided batch_ids

build(*id_map=None*)
Build the model based on provided list of ordered ids

3.6 Reader

@author: Quoc-Tuan Truong <tuantq.vnu@gmail.com>

cornac.data.reader.**read_ui**(*fpath*, *value=1.0*, *sep='\t'*, *skip_lines=0*)
Read data in the form of implicit feedback user-items. Each line starts with user id followed by multiple of item ids.

Parameters

- **fpath** (*str*) – Path to the data file
- **value** (*float, default: 1.0*) – Value for the feedback
- **sep** (*str, default:*) – The delimiter string.
- **skip_lines** (*int, default: 0*) – Number of first lines to skip

Returns **triplets** – Data in the form of list of tuples of (user, item, 1).

Return type iterable

cornac.data.reader.**read_uir**(*fpath*, *u_col=0*, *i_col=1*, *r_col=2*, *sep='\t'*, *skip_lines=0*)

Read data in the form of triplets (user, item, rating).

Parameters

- **fpath** (*str*) – Path to the data file
- **u_col** (*int, default: 0*) – Index of the user column
- **i_col** (*int, default: 1*) – Index of the item column
- **r_col** (*int, default: 2*) – Index of the rating column
- **sep** (*str, default:*) – The delimiter string.
- **skip_lines** (*int, default: 0*) – Number of first lines to skip

Returns **triplets** – Data in the form of list of tuples of (user, item, rating).

Return type iterable

CHAPTER 4

Models

4.1 Probabilistic Collaborative Representation Learning (PCRL)

@author: Aghiles Salah <asalah@smu.edu.sg>

class cornac.models.pcrl.recom_pcrl.**PCRL**(*k=100, z_dims=[300], max_iter=300, batch_size=300, learning_rate=0.001, name='pcrl', trainable=True, verbose=False, w_determinist=True, init_params={'G_r': None, 'G_s': None, 'L_r': None, 'L_s': None}*)

Probabilistic Collaborative Representation Learning.

Parameters

- **k** (*int, optional, default: 100*) – The dimension of the latent factors.
- **z_dims** (*Numpy 1d array, optional, default: [300]*) – The dimensions of the hidden intermdiate layers 'z' in the order [dim(z_L), ...,dim(z_1)], please refer to Figure 1 in the orginal paper for more details.
- **max_iter** (*int, optional, default: 300*) – Maximum number of iterations (number of epochs) for variational PCRL.
- **batch_size** (*int, optional, default: 300*) – The batch size for SGD.
- **learning_rate** (*float, optional, default: 0.001*) – The learning rate for SGD.
- **aux_info** (*see "cornac/examples/pcrl_example.py" in the GitHub repo for an example of how to use cornac's graph module provide item auxiliary data (e.g., context, text, etc.) for PCRL.*) –
- **name** (*string, optional, default: 'PCRL'*) – The name of the recommender model.

- **trainable** (*boolean, optional, default: True*) – When False, the model is not trained and Cornac assumes that the model already pre-trained (Theta, Beta and Xi are not None).
- **w_determinist** (*boolean, optional, default: True*) – When True, determinist wheights "W" are used for the generator network, otherwise "W" is stochastic as in the original paper.
- **init_params** (*dictionary, optional, default: {'G_s':None, 'G_r':None, 'L_s':None, 'L_r':None}*) – List of initial parameters, e.g., init_params = {'G_s':G_s, 'G_r':G_r, 'L_s':L_s, 'L_r':L_r}, where G_s and G_r are of type csc_matrix or np.array with the same shape as Theta, see below). They represent respectively the "shape" and "rate" parameters of Gamma distribution over Theta. It is the same for L_s, L_r and Beta.
- **Theta** (*csc_matrix, shape (n_users,k)*) – The expected user latent factors.
- **Beta** (*csc_matrix, shape (n_items,k)*) – The expected item latent factors.

References

- Salah, Aghiles, and Hady W. Lauw. Probabilistic Collaborative Representation Learning for Personalized Item Recommendation. In UAI 2018.

fit (*train_set*)

Fit the model to observations.

Parameters **train_set** (*object of type TrainSet, required*) – An object containing the user-item preference in csr scipy sparse format, as well as some useful attributes such as mappings to the original user/item ids. Please refer to the class TrainSet in the "data" module for details.

score (*user_id*, *item_id=None*)

Predict the scores/ratings of a user for a list of items.

Parameters

- **user_id** (*int, required*) – The index of the user for whom to perform score prediction.
- **item_id** (*int, optional, default: None*) – The index of the item for that to perform score prediction. If None, scores for all known items will be returned.

Returns **res** – Relative scores that the user gives to the item or to all known items

Return type A scalar or a Numpy array

4.2 Collaborative Context Poisson Factorization (C2PF)

@author: Aghiles Salah <asalah@smu.edu.sg>

class cornac.models.c2pf.recom_c2pf.**C2PF**(*k=100, max_iter=100, variant='c2pf', name=None, trainable=True, verbose=False, init_params={'G_r': None, 'G_s': None, 'L2_r': None, 'L2_s': None, 'L3_r': None, 'L3_s': None, 'L_r': None, 'L_s': None}*)

Collaborative Context Poisson Factorization.

Parameters

- **k** (*int, optional, default: 100*) – The dimension of the latent factors.
- **max_iter** (*int, optional, default: 100*) – Maximum number of iterations for variational C2PF.
- **variant** (*string, optional, default: 'c2pf'*) – C2pf's variant: c2pf: 'c2pf', 'tc2pf' (tied-c2pf) or 'rc2pf' (reduced-c2pf). Please refer to the original paper for details.
- **name** (*string, optional, default: None*) – The name of the recommender model. If None, then "variant" is used as the default name of the model.
- **trainable** (*boolean, optional, default: True*) – When False, the model is not trained and Cornac assumes that the model already pre-trained (Theta, Beta and Xi are not None).
- **Item_context** (*See "cornac/examples/c2pf_example.py" in the GitHub repo for an example of how to use cornac's graph module to load and provide "item context" for C2PF.*) –
- **init_params** (*dictionary, optional, default: {'G_s':None, 'G_r':None, 'L_s':None, 'L_r':None, 'L2_s':None, 'L2_r':None, 'L3_s':None, 'L3_r':None}*) – List of initial parameters, e.g., init_params = {'G_s':G_s, 'G_r':G_r, 'L_s':L_s, 'L_r':L_r, 'L2_s':L2_s, 'L2_r':L2_r, 'L3_s':L3_s, 'L3_r':L3_r}, where G_s and G_r are of type csc_matrix or np.array with the same shape as Theta, see below). They represent respectively the "shape" and "rate" parameters of Gamma distribution over Theta. It is the same for L_s, L_r and Beta, L2_s, L2_r and Xi, L3_s, L3_r and Kappa.
- **Theta** (*csc_matrix, shape (n_users,k)*) – The expected user latent factors.
- **Beta** (*csc_matrix, shape (n_items,k)*) – The expected item latent factors.
- **Xi** (*csc_matrix, shape (n_items,k)*) – The expected context item latent factors multiplied by context effects Kappa, please refer to the paper below for details.

References

- Salah, Aghiles, and Hady W. Lauw. A Bayesian Latent Variable Model of User Preferences with Item Context. In IJCAI, pp. 2667-2674. 2018.

fit (*train_set*)

Fit the model to observations.

Parameters **train_set** (*object of type TrainSet, required*) – An object containing the user-item preference in csr scipy sparse format, as well as some useful attributes such as mappings to the original user/item ids. Please refer to the class TrainSet in the "data" module for details.

score (*user_id*, *item_id=None*)

Predict the scores/ratings of a user for an item.

Parameters

- **user_id** (*int, required*) – The index of the user for whom to perform score prediction.
- **item_id** (*int, optional, default: None*) – The index of the item for that to perform score prediction. If None, scores for all known items will be returned.

Returns **res** – Relative scores that the user gives to the item or to all known items

Return type A scalar or a Numpy array

4.3 Indexable Bayesian Personalized Ranking (IBPR)

@author: Dung D. Le (Andrew) <ddle.2015@smu.edu.sg>

class cornac.models.ibpr.recom_ibpr.**IBPR**(*k=20*, *max_iter=100*, *learning_rate=0.05*, *lamda=0.001*, *batch_size=100*, *name='ibpr'*, *trainable=True*, *verbose=False*, *init_params=None*)

Indexable Bayesian Personalized Ranking.

Parameters

- **k** (*int, optional, default: 20*) – The dimension of the latent factors.
- **max_iter** (*int, optional, default: 100*) – Maximum number of iterations or the number of epochs for SGD.
- **learning_rate** (*float, optional, default: 0.05*) – The learning rate for SGD.
- **lamda** (*float, optional, default: 0.001*) – The regularization parameter.
- **batch_size** (*int, optional, default: 100*) – The batch size for SGD.
- **name** (*string, optional, default: 'IBRP'*) – The name of the recommender model.
- **trainable** (*boolean, optional, default: True*) – When False, the model is not trained and Cornac assumes that the model already pre-trained (U and V are not None).
- **verbose** (*boolean, optional, default: False*) – When True, some running logs are displayed.
- **init_params** (*dictionary, optional, default: None*) – List of initial parameters, e.g., init_params = {'U':U, 'V':V} please see below the definition of U and V.
- **U** (*csc_matrix, shape (n_users,k)*) – The user latent factors, optional initialization via init_params.
- **V** (*csc_matrix, shape (n_items,k)*) – The item latent factors, optional initialization via init_params.

References

- Le, D. D., & Lauw, H. W. (2017, November). Indexable Bayesian personalized ranking for efficient top-k recommendation. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (pp. 1389-1398). ACM.

fit(*train_set*)

Fit the model to observations.

Parameters **train_set** (*object of type TrainSet, required*) – An object containing the user-item preference in csr scipy sparse format, as well as some useful attributes such as mappings to the original user/item ids. Please refer to the class TrainSet in the "data" module for details.

score(*user_id*, *item_id=None*)

Predict the scores/ratings of a user for an item.

Parameters

- **user_id** (*int, required*) – The index of the user for whom to perform score prediction.
- **item_id** (*int, optional, default: None*) – The index of the item for that to perform score prediction. If None, scores for all known items will be returned.

Returns **res** – Relative scores that the user gives to the item or to all known items

Return type A scalar or a Numpy array

4.4 Online Indexable Bayesian Personalized Ranking (OIBPR)

@author: Dung D. Le (Andrew) <ddle.2015@smu.edu.sg>

class cornac.models.online_ibpr.recom_online_ibpr.**OnlineIBPR**(*k=20*, *max_iter=100*, *learning_rate=0.05*, *lamda=0.001*, *batch_size=100*, *name='online_ibpr'*, *trainable=True*, *verbose=False*, *init_params=None*)

Online Indexable Bayesian Personalized Ranking.

Parameters

- **k** (*int, optional, default: 20*) – The dimension of the latent factors.
- **max_iter** (*int, optional, default: 100*) – Maximum number of iterations or the number of epochs for SGD.
- **learning_rate** (*float, optional, default: 0.05*) – The learning rate for SGD.
- **lamda** (*float, optional, default: 0.001*) – The regularization parameter.
- **batch_size** (*int, optional, default: 100*) – The batch size for SGD.
- **name** (*string, optional, default: 'IBRP'*) – The name of the recommender model.
- **trainable** (*boolean, optional, default: True*) – When False, the model is not trained and Cornac assumes that the model already pre-trained (U and V are not None).
- **verbose** (*boolean, optional, default: False*) – When True, some running logs are displayed.

- **init_params** (*dictionary, optional, default: None*) – List of initial parameters, e.g., init_params = {'U':U, 'V':V} please see below the definition of U and V.
- **U** (*csc_matrix, shape (n_users,k)*) – The user latent factors, optional initialization via init_params.
- **V** (*csc_matrix, shape (n_items,k)*) – The item latent factors, optional initialization via init_params.

References

- Le, D. D., & Lauw, H. W. (2017, November). Indexable Bayesian personalized ranking for efficient top-k recommendation. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (pp. 1389-1398). ACM.

fit(*train_set*)

Fit the model to observations.

Parameters **train_set** (*object of type TrainSet, required*) – An object containing the user-item preference in csr scipy sparse format, as well as some useful attributes such as mappings to the original user/item ids. Please refer to the class TrainSet in the "data" module for details.

score(*user_id*, *item_id=None*)

Predict the scores/ratings of a user for an item.

Parameters

- **user_id** (*int, required*) – The index of the user for whom to perform score prediction.
- **item_id** (*int, optional, default: None*) – The index of the item for that to perform score prediction. If None, scores for all known items will be returned.

Returns **res** – Relative scores that the user gives to the item or to all known items

Return type A scalar or a Numpy array

4.5 Collaborative Ordinal Embedding (COE)

@author: Dung D. Le (Andrew) <ddle.2015@smu.edu.sg>

class cornac.models.coe.recom_coe.**COE**(*k=20*, *max_iter=100*, *learning_rate=0.05*, *lamda=0.001*, *batch_size=1000*, *name='coe'*, *trainable=True*, *verbose=False*, *init_params=None*)

Collaborative Ordinal Embedding.

Parameters

- **k** (*int, optional, default: 20*) – The dimension of the latent factors.
- **max_iter** (*int, optional, default: 100*) – Maximum number of iterations or the number of epochs for SGD.
- **learning_rate** (*float, optional, default: 0.05*) – The learning rate for SGD.
- **lamda** (*float, optional, default: 0.001*) – The regularization parameter.

- **batch_size** (*int, optional, default: 100*) – The batch size for SGD.
- **name** (*string, optional, default: 'IBRP'*) – The name of the recommender model.
- **trainable** (*boolean, optional, default: True*) – When False, the model is not trained and Cornac assumes that the model already pre-trained (U and V are not None).
- **verbose** (*boolean, optional, default: False*) – When True, some running logs are displayed.
- **init_params** (*dictionary, optional, default: None*) – List of initial parameters, e.g., init_params = {'U':U, 'V':V} please see below the definition of U and V.
- **U** (*csc_matrix, shape (n_users,k)*) – The user latent factors, optional initialization via init_params.
- **V** (*csc_matrix, shape (n_items,k)*) – The item latent factors, optional initialization via init_params.

References

- Le, D. D., & Lauw, H. W. (2016, June). Euclidean co-embedding of ordinal data for multi-type visualization. In Proceedings of the 2016 SIAM International Conference on Data Mining (pp. 396-404). Society for Industrial and Applied Mathematics.

fit (*train_set*)

Fit the model to observations.

Parameters **train_set** (*object of type TrainSet, required*) – An object containing the user-item preference in csr scipy sparse format, as well as some useful attributes such as mappings to the original user/item ids. Please refer to the class TrainSet in the "data" module for details.

score (*user_id*, *item_id=None*)

Predict the scores/ratings of a user for an item.

Parameters

- **user_id** (*int, required*) – The index of the user for whom to perform score prediction.
- **item_id** (*int, optional, default: None*) – The index of the item for that to perform score prediction. If None, scores for all known items will be returned.

Returns **res** – Relative scores that the user gives to the item or to all known items

Return type A scalar or a Numpy array

4.6 Visual Bayesian Personalized Ranking (VBPR)

@author: Guo Jingyao <jyguo@smu.edu.sg> Quoc-Tuan Truong <tuantq.vnu@gmail.com>

class cornac.models.vbpr.recom_vbpr.**VBPR**(*k=10*, *k2=10*, *n_epochs=20*, *batch_size=100*, *learning_rate=0.001*, *lambda_w=0.01*, *lambda_b=0.01*, *lambda_e=0.0*, *use_gpu=False*, *trainable=True*, *init_params=None*, ***kwargs*)

Visual Bayesian Personalized Ranking.

Parameters

- **k** (*int, optional, default: 10*) – The dimension of the gamma latent factors.
- **k2** (*int, optional, default: 10*) – The dimension of the theta latent factors.
- **n_epochs** (*int, optional, default: 20*) – Maximum number of epochs for SGD.
- **batch_size** (*int, optional, default: 100*) – The batch size for SGD.
- **learning_rate** (*float, optional, default: 0.001*) – The learning rate for SGD.
- **lambda_w** (*float, optional, default: 0.01*) – The regularization hyper-parameter for latent factor weights.
- **lambda_b** (*float, optional, default: 0.01*) – The regularization hyper-parameter for biases.
- **lambda_e** (*float, optional, default: 0.0*) – The regularization hyper-parameter for embedding matrix E and beta prime vector.
- **use_gpu** (*boolean, optional, default: True*) – Whether or not to use GPU to speed up training.
- **trainable** (*boolean, optional, default: True*) – When False, the model is not trained and Cornac assumes that the model already pre-trained (U and V are not None).
- **init_params** (*dictionary, optional, default: None*) –

  Initial parameters, e.g., init_params = {'Bi': beta_item, 'Gu': gamma_user, 'Gi': gamma_item, 'Tu': theta_user, 'E': emb_matrix, 'Bp': beta_prime}

References

- HE, Ruining et MCAULEY, Julian. VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback. In : AAAI. 2016. p. 144-150.

fit(*train_set*)

Fit the model.

Parameters **train_set** (*cornac.data.MultimodalTrainSet*) – Multimodal training set.

score(*user_id*, *item_id=None*)

Predict the scores/ratings of a user for an item.

Parameters

- **user_id** (*int, required*) – The index of the user for whom to perform score prediction.
- **item_id** (*int, optional, default: None*) – The index of the item for that to perform score prediction. If None, scores for all known items will be returned.

Returns **res** – Relative scores that the user gives to the item or to all known items

Return type A scalar or a Numpy array

4.7 Spherical k-means (Skmeans)

@author: Aghiles Salah <asalah@smu.edu.sg>

class cornac.models.skm.recom_skmeans.**SKMeans**(*k=5*, *max_iter=100*, *name='Skmeans'*, *trainable=True*, *tol=1e-06*, *verbose=True*, *init_par=None*)

Spherical k-means based recommender.

Parameters

- **k** (*int, optional, default: 5*) – The number of clusters.
- **max_iter** (*int, optional, default: 100*) – Maximum number of iterations.
- **name** (*string, optional, default: 'Skmeans'*) – The name of the recommender model.
- **trainable** (*boolean, optional, default: True*) – When False, the model is not trained and Cornac assumes that the model is already trained.
- **tol** (*float, optional, default: 1e-6*) – Relative tolerance with regards to skmeans' criterion to declare convergence.
- **verbose** (*boolean, optional, default: False*) – When True, some running logs are displayed.
- **init_par** (*numpy 1d array, optional, default: None*) – The initial object parition, 1d array contaning the cluster label (int type starting from 0) of each object (user). If par = None, then skmeans is initialized randomly.
- **centroids** (*csc_matrix, shape (k,n_users)*) – The maxtrix of cluster centroids.

References

- Salah, Aghiles, Nicoleta Rogovschi, and Mohamed Nadif. "A dynamic collaborative filtering system via a weighted clustering approach." Neurocomputing 175 (2016): 206-215.

fit(*train_set*)

Fit the model to observations.

Parameters **train_set** (*object of type TrainSet, required*) – An object containing the user-item preference in csr scipy sparse format, as well as some useful attributes such as mappings to the original user/item ids. Please refer to the class TrainSet in the "data" module for details.

score(*user_id*, *item_id=None*)

Predict the scores/ratings of a user for an item.

Parameters

- **user_id** (*int, required*) – The index of the user for whom to perform score prediction.

- **item_id** (*int, optional, default: None*) – The index of the item for that to perform score prediction. If None, scores for all known items will be returned.

Returns **res** – Relative scores that the user gives to the item or to all known items

Return type A scalar or a Numpy array

4.8 Collaborative Deep Learning (CDL)

@author: Trieu Thi Ly Ly

class cornac.models.cdl.recom_cdl.**CDL**(*k=50*, *text_information=None*, *autoencoder_structure=None*, *lambda_u=0.1*, *lambda_v=0.01*, *lambda_w=0.01*, *lambda_n=0.01*, *a=1*, *b=0.01*, *autoencoder_corruption=0.3*, *learning_rate=0.001*, *keep_prob=1.0*, *batch_size=100*, *max_iter=100*, *name='CDL'*, *trainable=True*, *verbose=False*, *init_params=None*)

Collaborative Deep Learning.

Parameters

- **k** (*int, optional, default: 50*) – The dimension of the latent factors.
- **max_iter** (*int, optional, default: 100*) – Maximum number of iterations or the number of epochs for SGD.
- **shape (n_items, n_vocabularies), optional, default** (*text_informationndarray,*) – Bag-of-words features of items
- **optional, default** (*autoencoder_structurearray,*) – The number of neurons of encoder/ decoder layer for SDAE
- **learning_rate** (*float, optional, default: 0.001*) – The learning rate for AdamOptimizer.
- **lambda_u** (*float, optional, default: 0.1*) – The regularization parameter for users.
- **lambda_v** (*float, optional, default: 10*) – The regularization parameter for items.
- **lambda_w** (*float, optional, default: 0.1*) – The regularization parameter for SDAE weights.
- **lambda_n** (*float, optional, default: 1000*) – The regularization parameter for SDAE output.
- **a** (*float, optional, default: 1*) – The confidence of observed ratings.
- **b** (*float, optional, default: 0.01*) – The confidence of unseen ratings.
- **autoencoder_corruption** (*float, optional, default: 0.3*) – The corruption ratio for SDAE.
- **keep_prob** (*float, optional, default: 1.0*) – The probability that each element is kept in dropout of SDAE.
- **batch_size** (*int, optional, default: 100*) – The batch size for SGD.
- **name** (*string, optional, default: 'CDL'*) – The name of the recommender model.

- **trainable** (*boolean, optional, default: True*) – When False, the model is not trained and Cornac assumes that the model already pre-trained (U and V are not None).
- **init_params** (*dictionary, optional, default: None*) – List of initial parameters, e.g., init_params = {'U':U, 'V':V} please see below the definition of U and V.
- **U** (*ndarray, shape (n_users,k)*) – The user latent factors, optional initialization via init_params.
- **V** (*ndarray, shape (n_items,k)*) – The item latent factors, optional initialization via init_params.

References

- Hao Wang, Naiyan Wang, Dit-Yan Yeung. CDL: Collaborative Deep Learning for Recommender Systems. In : SIGKDD. 2015. p. 1235-1244.

fit (*train_set*)

Fit the model to observations.

Parameters **train_set** (*object of type TrainSet, required*) – An object containing the user-item preference in csr scipy sparse format, as well as some useful attributes such as mappings to the original user/item ids. Please refer to the class TrainSet in the "data" module for details.

score (*user_id*, *item_id=None*)

Predict the scores/ratings of a user for an item.

Parameters

- **user_id** (*int, required*) – The index of the user for whom to perform score prediction.
- **item_id** (*int, optional, default: None*) – The index of the item for that to perform score prediction. If None, scores for all known items will be returned.

Returns **res** – Relative scores that the user gives to the item or to all known items

Return type A scalar or a Numpy array

4.9 Hierarchical Poisson Factorization (HPF)

@author: Aghiles Salah <asalah@smu.edu.sg>

class cornac.models.hpf.recom_hpf.**HPF**(*k=5*, *max_iter=100*, *name='HPF'*, *trainable=True*, *verbose=False*, *hierarchical=True*, *init_params={'G_r': None, 'G_s': None, 'L_r': None, 'L_s': None}*)

Hierarchical Poisson Factorization.

Parameters

- **k** (*int, optional, default: 5*) – The dimension of the latent factors.
- **max_iter** (*int, optional, default: 100*) – Maximum number of iterations.

- **name** (*string, optional, default: 'HPF'*) – The name of the recommender model.
- **trainable** (*boolean, optional, default: True*) – When False, the model is not trained and Cornac assumes that the model is already pre-trained (Theta and Beta are not None).
- **verbose** (*boolean, optional, default: False*) – When True, some running logs are displayed.
- **hierarchical** (*boolean, optional, default: True*) – When False, PF is used instead of HPF.
- **init_params** (*dictionary, optional, default: {'G_s':None, 'G_r':None, 'L_s':None, 'L_r':None}*) – List of initial parameters, e.g., init_params = {'G_s':G_s, 'G_r':G_r, 'L_s':L_s, 'L_r':L_r}, where G_s and G_r are of type csc_matrix or np.array with the same shape as Theta, see below). They represent respectively the "shape" and "rate" parameters of Gamma distribution over Theta. Similarly, L_s, L_r are the shape and rate parameters of the Gamma over Beta.
- **Theta** (*csc_matrix, shape (n_users,k)*) – The expected user latent factors.
- **Beta** (*csc_matrix, shape (n_items,k)*) – The expected item latent factors.

References

- Gopalan, Prem, Jake M. Hofman, and David M. Blei. Scalable Recommendation with Hierarchical Poisson Factorization. In UAI, pp. 326-335. 2015.

fit (*train_set*)
Fit the model to observations.

Parameters **train_set** (*object of type TrainSet, required*) – An object containing the user-item preference in csr scipy sparse format, as well as some useful attributes such as mappings to the original user/item ids. Please refer to the class TrainSet in the "data" module for details.

score (*user_id*, *item_id=None*)
Predict the scores/ratings of a user for an item.

Parameters

- **user_id** (*int, required*) – The index of the user for whom to perform score prediction.
- **item_id** (*int, optional, default: None*) – The index of the item for that to perform score prediction. If None, scores for all known items will be returned.

Returns **res** – Relative scores that the user gives to the item or to all known items

Return type A scalar or a Numpy array

4.10 Bayesian Personalized Ranking (BPR)

class cornac.models.bpr.recom_bpr.**BPR**
Bayesian Personalized Ranking.

Parameters

- **k** (*int, optional, default: 10*) – The dimension of the latent factors.
- **max_iter** (*int, optional, default: 100*) – Maximum number of iterations or the number of epochs for SGD.
- **learning_rate** (*float, optional, default: 0.001*) – The learning rate for SGD.
- **lambda_reg** (*float, optional, default: 0.001*) – The regularization hyper-parameter.
- **num_threads** (*int, optional, default: 0*) – Number of parallel threads for training. If 0, all CPU cores will be utilized.
- **trainable** (*boolean, optional, default: True*) – When False, the model will not be re-trained, and input of pre-trained parameters are required.
- **verbose** (*boolean, optional, default: True*) – When True, some running logs are displayed.
- **init_params** (*dictionary, optional, default: None*) – Initial parameters, e.g., init_params = {'U': user_factors, 'V': item_factors, 'Bi': item_biases}

References

- Rendle, Steffen, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In UAI, pp. 452-461. 2009.

fit

Fit the model to observations.

Parameters **train_set** (*object of type TrainSet, required*) – An object contains the user-item preference in csr scipy sparse format, as well as some useful attributes such as mappings to the original user/item ids. Please refer to the class TrainSet in the "data" module for details.

score

Predict the scores/ratings of a user for an item.

Parameters

- **user_id** (*int, required*) – The index of the user for whom to perform score prediction.
- **item_id** (*int, optional, default: None*) – The index of the item for that to perform score prediction. If None, scores for all known items will be returned.

Returns **res** – Relative scores that the user gives to the item or to all known items

Return type A scalar or a Numpy array

4.11 Probabilitic Matrix Factorization (PMF)

@author: Aghiles Salah

class cornac.models.pmf.recom_pmf.**PMF**(*k=5*, *max_iter=100*, *learning_rate=0.001*, *gamma=0.9*, *lamda=0.001*, *name='PMF'*, *variant='non_linear'*, *trainable=True*, *verbose=False*, *init_params={'U': None, 'V': None}*)

Probabilistic Matrix Factorization.

Parameters

- **k** (*int, optional, default: 5*) – The dimension of the latent factors.
- **max_iter** (*int, optional, default: 100*) – Maximum number of iterations or the number of epochs for SGD.
- **learning_rate** (*float, optional, default: 0.001*) – The learning rate for SGD_RMSProp.
- **gamma** (*float, optional, default: 0.9*) – The weight for previous/current gradient in RMSProp.
- **lamda** (*float, optional, default: 0.001*) – The regularization parameter.
- **name** (*string, optional, default: 'PMF'*) – The name of the recommender model.
- **variant** (*{"linear","non_linear"}, optional, default: 'non_linear'*) – Pmf variant. If 'non_linear', the Gaussian mean is the output of a Sigmoid function. If 'linear' the Gaussian mean is the output of the identity function.
- **trainable** (*boolean, optional, default: True*) – When False, the model is not trained and Cornac assumes that the model already pre-trained (U and V are not None).
- **verbose** (*boolean, optional, default: False*) – When True, some running logs are displayed.
- **init_params** (*dictionary, optional, default: {'U':None, 'V':None}*) – List of initial parameters, e.g., init_params = {'U':U, 'V':V}. U: a csc_matrix of shape (n_users,k), containing the user latent factors. V: a csc_matrix of shape (n_items,k), containing the item latent factors.

References

- Mnih, Andriy, and Ruslan R. Salakhutdinov. Probabilistic matrix factorization. In NIPS, pp. 1257-1264. 2008.

fit(*train_set*)

Fit the model to observations.

Parameters **train_set** (*object of type TrainSet, required*) – An object containing the user-item preference in csr scipy sparse format, as well as some useful attributes such as mappings to the original user/item ids. Please refer to the class TrainSet in the "data" module for details.

score(*user_id*, *item_id=None*)

Predict the scores/ratings of a user for an item.

Parameters

- **user_id** (*int, required*) – The index of the user for whom to perform score prediction.

- **item_id** (*int, optional, default: None*) – The index of the item for that to perform score prediction. If None, scores for all known items will be returned.

Returns **res** – Relative scores that the user gives to the item or to all known items

Return type A scalar or a Numpy array

4.12 Matrix Factorization (MF)

@author: Quoc-Tuan Truong <tuantq.vnu@gmail.com>

class cornac.models.mf.recom_mf.**MF**

Matrix Factorization.

Parameters

- **k** (*int, optional, default: 10*) – The dimension of the latent factors.
- **max_iter** (*int, optional, default: 100*) – Maximum number of iterations or the number of epochs for SGD.
- **learning_rate** (*float, optional, default: 0.01*) – The learning rate.
- **lambda_reg** (*float, optional, default: 0.001*) – The lambda value used for regularization.
- **use_bias** (*boolean, optional, default: True*) – When True, user, item, and global biases are used.
- **early_stop** (*boolean, optional, default: False*) – When True, delta loss will be checked after each iteration to stop learning earlier.
- **trainable** (*boolean, optional, default: True*) – When False, the model will not be re-trained, and input of pre-trained parameters are required.
- **verbose** (*boolean, optional, default: True*) – When True, running logs are displayed.
- **init_params** (*dictionary, optional, default: None*) – Initial parameters, e.g., init_params = {'U': user_factors, 'V': item_factors, 'Bu': user_biases, 'Bi': item_biases}

References

- Koren, Y., Bell, R., & Volinsky, C. Matrix factorization techniques for recommender systems. In Computer, (8), 30-37. 2009.

fit

Fit the model to observations.

Parameters **train_set** (*object of type TrainSet, required*) – An object contains the user-item preference in csr scipy sparse format, as well as some useful attributes such as mappings to the original user/item ids. Please refer to the class TrainSet in the "data" module for details.

score

Predict the scores/ratings of a user for an item.

Parameters

- **user_id** (*int, required*) – The index of the user for whom to perform score prediction.
- **item_id** (*int, optional, default: None*) – The index of the item for that to perform score prediction. If None, scores for all known items will be returned.

Returns **res** – Relative scores that the user gives to the item or to all known items

Return type A scalar or a Numpy array

4.13 Convolutional Matrix Factorization (ConvMF)

@author: Tran Thanh Binh

class cornac.models.conv_mf.recom_convmf.**ConvMF**(*give_item_weight=True*, *n_epochs=50*, *lambda_u=1*, *lambda_v=100*, *k=50*, *name='convmf'*, *trainable=True*, *verbose=False*, *dropout_rate=0.2*, *emb_dim=200*, *max_len=300*, *num_kernel_per_ws=100*, *init_params=None*)

Parameters

- **k** (*int, optional, default: 50*) – The dimension of the user and item latent factors.
- **n_epochs** (*int, optional, default: 50*) – Maximum number of epochs for training.
- **lambda_u** (*float, optional, default: 1.0*) – The regularization hyper-parameter for user latent factor.
- **lambda_v** (*float, optional, default: 100.0*) – The regularization hyper-parameter for item latent factor.
- **emb_dim** (*int, optional, default: 200*) – The embedding size of each word. One word corresponds with [1 x emb_dim] vector in the embedding space
- **max_len** (*int, optional, default 300*) – The maximum length of item's document
- **num_kernel_per_ws** (*int, optional, default: 100*) – The number of kernel filter in convolutional layer
- **dropout_rate** (*float, optional, default: 0.2*) – Dropout rate while training CNN
- **give_item_weight** (*boolean, optional, default: True*) – When True, each item will be weighted base on the number of user who have rated this item
- **init_params** (*dict, optional, default: {'U':None, 'V':None, 'W': None}*) – Initial U and V matrix and initial weight for embedding layer W
- **trainable** (*boolean, optional, default: True*) – When False, the model is not trained and Cornac assumes that the model already pre-trained (U and V are not None).

References

- Donghyun Kim1, Chanyoung Park1. ConvMF: Convolutional Matrix Factorization for Document Context-Aware Recommendation. In :10th ACM Conference on Recommender Systems Pages 233-240

fit(*train_set*)
Fit the model.

Parameters **train_set** (*cornac.data.MultimodalTrainSet*) – Multimodal training set.

score(*user_id*, *item_id=None*)
Predict the scores/ratings of a user for an item.

Parameters

- **user_id** (*int, required*) – The index of the user for whom to perform score prediction.
- **item_id** (*int, optional, default: None*) – The index of the item for that to perform score prediction. If None, scores for all known items will be returned.

Returns **res** – Relative scores that the user gives to the item or to all known items

Return type A scalar or a Numpy array

CHAPTER 5

Metrics

5.1 Area Under the Curve (AUC)

class cornac.metrics.**AUC**
Area Under the ROC Curve (AUC).

References

https://arxiv.org/ftp/arxiv/papers/1205/1205.2618.pdf

5.2 Normalized Discount Cumulative Gain (NDCG)

class cornac.metrics.**NDCG**(*k=-1*)
Normalized Discount Cumulative Gain.

Parameters **k** (*int, optional, default: -1 (all)*) – The number of items in the top@k list. If None, all items will be considered.

References

https://en.wikipedia.org/wiki/Discounted_cumulative_gain

5.3 Normalized Cumulative Reciprocal Rank (NCRR)

class cornac.metrics.**NCRR**(*k=-1*)
Normalized Cumulative Reciprocal Rank.

Parameters **k** (*int, optional, default: -1 (all)*) – The number of items in the top@k list. If None, all items will be considered.

5.4 Mean Reciprocal Rank (MRR)

class cornac.metrics.**MRR**
Mean Reciprocal Rank.

Parameters **k** (*int, optional, default: -1 (all)*) – The number of items in the top@k list. If None, all items will be considered.

References

https://en.wikipedia.org/wiki/Mean_reciprocal_rank

5.5 Precision

class cornac.metrics.**Precision**(*k=-1*)
Precision@K.

Parameters **k** (*int, optional, default: -1 (all)*) – The number of items in the top@k list. If None, all items will be considered.

5.6 Recall

class cornac.metrics.**Recall**(*k=-1*)
Recall@K.

Parameters **k** (*int, optional, default: -1 (all)*) – The number of items in the top@k list. If None, all items will be considered.

5.7 Fmeasure (F1)

class cornac.metrics.**FMeasure**(*k=-1*)
F-measure@K@.

Parameters **k** (*int, optional, default: -1 (all)*) – The number of items in the top@k list. If None, all items will be considered.

5.8 Mean Absolute Error (MAE)

class cornac.metrics.**MAE**
Mean Absolute Error.

name
Name of the measure.

Type string, value: 'MAE'

5.9 Root Mean Squared Error (RMSE)

class cornac.metrics.**RMSE**
Root Mean Squared Error.

name
Name of the measure.

Type string, value: 'RMSE'

CHAPTER 6

Evaluation methods

6.1 Base Method

@author: Quoc-Tuan Truong <tuantq.vnu@gmail.com>

class cornac.eval_methods.base_method.**BaseMethod**(*data=None*, *fmt='UIR'*, *rating_threshold=1.0*, *exclude_unknowns=False*, *verbose=False*, ***kwargs*)

Base Evaluation Method

Parameters

- **data** (*array-like*) – The original data.
- **data_format** (*str, default: 'UIR'*) – The format of given data.
- **total_users** (*int, optional, default: None*) – Total number of unique users in the data including train, val, and test sets.
- **total_users** – Total number of unique items in the data including train, val, and test sets.
- **rating_threshold** (*float, optional, default: 1.0*) – The threshold to convert ratings into positive or negative feedback for ranking metrics.
- **exclude_unknowns** (*bool, optional, default: False*) – Ignore unknown users and items (cold-start) during evaluation.
- **verbose** (*bool, optional, default: False*) – Output running log

evaluate(*model*, *metrics*, *user_based*)

Evaluate given models according to given metrics

Parameters

- **model** (cornac.models.Recommender) – Recommender model to be evaluated.
- **metrics** (iterable) – List of metrics.

- **user_based** (*bool*) – Evaluation mode. Whether results are averaging based on number of users or number of ratings.

classmethod from_splits(*train_data*, *test_data*, *val_data=None*, *data_format='UIR'*, *rating_threshold=1.0*, *exclude_unknowns=False*, *verbose=False*)

Constructing evaluation method given data.

Parameters

- **train_data** (*array-like*) – Training data
- **test_data** (*array-like*) – Test data
- **val_data** (*array-like*) – Validation data
- **data_format** (*str, default: 'UIR'*) – The format of given data.
- **rating_threshold** (*float, default: 1.0*) – Threshold to decide positive or negative preferences.
- **exclude_unknowns** (*bool, default: False*) – Whether to exclude unknown users/items in evaluation.
- **verbose** (*bool, default: False*) – The verbosity flag.

Returns **method** – Evaluation method object.

Return type <cornac.eval_methods.BaseMethod>

6.2 Ratio Split

@author: Quoc-Tuan Truong <tuantq.vnu@gmail.com>

class cornac.eval_methods.ratio_split.**RatioSplit**(*data*, *fmt='UIR'*, *test_size=0.2*, *val_size=0.0*, *rating_threshold=1.0*, *shuffle=True*, *seed=None*, *exclude_unknowns=False*, *verbose=False*, ***kwargs*)

Train-Test Split Evaluation Method.

Parameters

- **data** (*.., required*) – The input data in the form of triplets (user, item, rating).
- **fmt** (*str, optional, default: "UIR"*) – The format of input data: - UIR: (user, item, rating) triplet data - UIRT: (user, item , rating, timestamp) quadruplet data
- **test_size** (*float, optional, default: 0.2*) – The proportion of the test set, if > 1 then it is treated as the size of the test set.
- **val_size** (*float, optional, default: 0.0*) – The proportion of the validation set, if > 1 then it is treated as the size of the validation set.
- **rating_threshold** (*float, optional, default: 1.*) – The minimum value that is considered to be a good rating used for ranking, e.g, if the ratings are in {1, . . . , 5}, then rating_threshold = 4.
- **shuffle** (*bool, optional, default: True*) – Shuffle the data before splitting.
- **seed** (*bool, optional, default: None*) – Random seed.

- **exclude_unknowns** (*bool, optional, default: False*) – Ignore unknown users and items (cold-start) during evaluation and testing
- **verbose** (*bool, optional, default: False*) – Output running log

evaluate (*model*, *metrics*, *user_based*)
Evaluate given models according to given metrics

Parameters

- **model** (cornac.models.Recommender) – Recommender model to be evaluated.
- **metrics** (iterable) – List of metrics.
- **user_based** (*bool*) – Evaluation mode. Whether results are averaging based on number of users or number of ratings.

6.3 Cross Validation

@author: Aghiles Salah

class cornac.eval_methods.cross_validation.**CrossValidation**(*data*, *fmt='UIR'*, *n_folds=5*, *rating_threshold=1.0*, *partition=None*, *exclude_unknowns=True*, *verbose=False*, ***kwargs*)

Cross Validation Evaluation Method.

Parameters

- **data** (*.. , required*) – Input data in the triplet format (user_id, item_id, rating_val).
- **n_folds** (*int, optional, default: 5*) – The number of folds for cross validation.
- **rating_threshold** (*float, optional, default: 1.*) – The minimum value that is considered to be a good rating, e.g, if the ratings are in {1, ... ,5}, then rating_threshold = 4.
- **partition** (*array-like, shape (n_observed_ratings,), optional, default: None*) – The partition of ratings into n_folds (fold label of each rating) If None, random partitioning is performed to assign each rating into a fold.
- **rating_threshold** – The minimum value that is considered to be a good rating used for ranking, e.g, if the ratings are in {1, ..., 5}, then rating_threshold = 4.
- **exclude_unknowns** (*bool, optional, default: False*) – Ignore unknown users and items (cold-start) during evaluation and testing
- **verbose** (*bool, optional, default: False*) – Output running log

evaluate (*model*, *metrics*, *user_based*)
Evaluate given models according to given metrics

Parameters

- **model** (cornac.models.Recommender) – Recommender model to be evaluated.
- **metrics** (iterable) – List of metrics.

- **user_based** (*bool*) – Evaluation mode. Whether results are averaging based on number of users or number of ratings.

CHAPTER 7

Experiment

class cornac.experiment.**Experiment**(*eval_method*, *models*, *metrics*, *user_based=True*, *verbose=False*)

Experiment Class

Parameters

- **eval_method** (*BaseMethod object, required*) – The evaluation method (e.g., RatioSplit).
- **models** (*array of objects Recommender, required*) – A collection of recommender models to evaluate, e.g., [C2pf, Hpf, Pmf].
- **metrics** (*array of object metrics, required*) – A collection of metrics to use to evaluate the recommender models, e.g., [Ndcg, Mrr, Recall].
- **user_based** (*bool, optional, default: True*) – Performance will be averaged based on number of users for rating metrics. If *False*, results will be averaged over number of ratings.
- **avg_results** (*DataFrame, default: None*) – The average result per model.
- **user_results** (*dictionary, default: {}*) – Results per user for each model. Result of user u, of metric m, of model d will be user_results[d][m][u]

CHAPTER 8

Built-in datasets

8.1 MovieLens

@author: Quoc-Tuan Truong <tuantq.vnu@gmail.com>

MovieLens: https://grouplens.org/datasets/movielens/

`cornac.datasets.movielens.load_100k(fmt='UIR')`
Load the MovieLens 100K dataset

Parameters **fmt** (*str, default: 'UIR'*) – Data format to be returned.

Returns **data** – Data in the form of a list of tuples depending on the given data format.

Return type array-like

`cornac.datasets.movielens.load_1m(fmt='UIR')`
Load the MovieLens 1M dataset

Parameters **fmt** (*str, default: 'UIR'*) – Data format to be returned.

Returns **data** – Data in the form of a list of tuples depending on the given data format.

Return type array-like

`cornac.datasets.movielens.load_plot()`
Load the plots of movies provided @ http://dm.postech.ac.kr/~cartopy/ConvMF/

Returns **movie_plots** – A dictionary with keys are movie ids and values are text plots.

Return type Dict

8.2 Netflix

@author: Quoc-Tuan Truong <tuantq.vnu@gmail.com>

Data: https://www.kaggle.com/netflix-inc/netflix-prize-data/

cornac.datasets.netflix.**load_data**(*fmt='UIR'*)
Load the Netflix entire dataset - Number of ratings: 100,480,507 - Number of users: 480,189 - Number of items: 17,770

Parameters **fmt** (*str, default: 'UIR'*) – Data format to be returned.

Returns **data** – Data in the form of a list of tuples depending on the given data format.

Return type array-like

cornac.datasets.netflix.**load_data_small**(*fmt='UIR'*)
Load a small subset of the Netflix dataset. We draw this subsample such that every user has at least 10 items and each item has at least 10 users. - Number of ratings: 607,803 - Number of users: 10,000 - Number of items: 5,000

Parameters **fmt** (*str, default: 'UIR'*) – Data format to be returned.

Returns **data** – Data in the form of a list of tuples depending on the given data format.

Return type array-like

8.3 Tradesy

@author: Quoc-Tuan Truong <tuantq.vnu@gmail.com>

Original data: http://jmcauley.ucsd.edu/data/tradesy/ This data is used in the VBPR paper. After cleaning the data, we have: - Number of feedback: 394,421 (410,186 is reported but there are duplicates) - Number of users: 19,243 (19,823 is reported due to duplicates) - Number of items: 165,906 (166,521 is reported due to duplicates)

cornac.datasets.tradesy.**load_data**()
Load the feedback observations

Returns **data** – Data in the form of a list of tuples (user, item, 1).

Return type array-like

cornac.datasets.tradesy.**load_feature**()
Load the item visual feature

Returns **data** – Item-feature dictionary. Each feature vector is a Numpy array of size 4096.

Return type dict

8.4 Amazon Office

@author: Aghiles Salah <asalah@smu.edu.sg>

This data is built based on the Amazon datasets provided by Julian McAuley at: http://jmcauley.ucsd.edu/data/amazon/

cornac.datasets.amazon_office.**load_context**(*data_format='UIR'*)
Load the item-item interactions

Parameters **data_format** (*str, default: 'UIR'*) – Data format to be returned.

Returns **data** – Data in the form of a list of tuples depending on the specified data format.

Return type array-like

cornac.datasets.amazon_office.**load_rating**(*data_format='UIR'*)
Load the user-item ratings

Parameters **data_format** (*str, default: 'UIR'*) – Data format to be returned.

Returns **data** – Data in the form of a list of tuples depending on the specified data format.

Return type array-like

Python Module Index

C

Index

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V