Cornac Documentation

Release 1.2.2

Cornac Contributors

Nov 29, 2019

Getting Started

Cornac is a comparative framework for multimodal recommender systems. It focuses on making it **convenient** to work with models leveraging **auxiliary data** (e.g., item descriptive text and image, social network, etc). **Cornac** enables **fast** experiments and **straightforward** implementations of new models. It is **highly compatible** with existing machine learning libraries (e.g., TensorFlow, PyTorch).

CHAPTER 1

Data

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

Modality that contains features in general

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.**TextModality**(*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_doc_freq: int = 1*, *tfidf_params: Dict = None*, ***kwargs*)

Text modality

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** (*float in range [0.0, 1.0] or int, default=1.0*) – When building the vocabulary ignore terms that have a document frequency strictly higher than the given threshold (corpus-specific stop words). If float, the value represents a proportion of documents, int for absolute counts. If *vocab* is not None, this will be ignored.
- **min_doc_freq** (*float in range [0.0, 1.0] or int, default=1*) – When building the vocabulary ignore terms that have a document frequency strictly lower than the given threshold. This value is also called cut-off in the literature. If float, the value represents a proportion of documents, int absolute counts. If *vocab* is not None, this will be ignored.
- **tfidf_params** (*dict or None, optional, default=None*) – If *None*, a default arguments of `<cornac.data.text.IfidfVectorizer>` will be used. List of parameters:

  'binary' [boolean, default=False] If True, all non zero counts are set to 1.

  'norm' ['l1', 'l2' or None, optional, default='l2'] Each output row will have unit norm, either: * 'l2': Sum of squares of vector elements is 1. The cosine similarity between two vectors is their dot product when l2 norm has been applied. * 'l1': Sum of absolute values of vector elements is 1. See `utils.common.normalize()`

  'use_idf' [boolean, default=True] Enable inverse-document-frequency reweighting.

  'smooth_idf' [boolean, default=True] Smooth idf weights by adding one to document frequencies, as if an extra document was seen containing every term in the collection exactly once. Prevents zero divisions.

  'sublinear_tf' [boolean (default=False)] Apply sublinear tf scaling, i.e. replace tf with 1 + log(tf).

batch_seq(*batch_ids*, *max_length=None*)

Return a numpy matrix of text sequences containing token ids with size=(len(batch_ids), max_length).

Parameters

- **batch_ids** (*Union[List, numpy.array], required*) – An array containing the ids of rows of text sequences to be returned.
- **max_length** (*int, optional*) – Cut-off length of returned sequences. If *None*, it will be inferred based on retrieved sequences.

Returns **batch_sequences** – Batch of sequences with zero-padding at the end.

Return type numpy.ndarray

batch_tfidf(*batch_ids*, *keep_sparse=False*)

Return matrix of TF-IDF features corresponding to provided batch_ids

Parameters

- **batch_ids** (*array*) – An array of ids to retrieve the corresponding features.
- **keep_sparse** (*bool, default = False*) – If *True*, the return feature matrix will be a *scipy.sparse.csr_matrix*. Otherwise, it will be a dense matrix.

Returns **batch_tfidf** – Batch of TF-IDF representations corresponding to input *batch_ids*.

Return type numpy.ndarray

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

Parameters **id_map** (*dict, optional*) – A dictionary holds mapping from original ids to mapped integer indices of users/items.

Returns **text_modality** – An object of type *TextModality*.

Return type <cornac.data.TextModality>

tfidf_matrix
Return tf-idf matrix.

class cornac.data.**ImageModality**(***kwargs*)
Image modality

Parameters

- **images** (*Union[List, numpy.ndarray], optional*) – A list or tensor of images that the row indices are aligned with user/item in *ids*.
- **paths** (*List[str], optional*) – A list of paths, to images stored on disk, which the row indices are aligned with user/item in *ids*..

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

Parameters

- **batch_ids** (*Union[List, numpy.array], required*) – An array containing the ids of rows of images to be returned.
- **target_size** (*tuple, optional, default: (256, 256)*) – Size (width, height) of returned images to be resized.
- **color_mode** (*str, optional, default: 'rgb'*) – Color mode of returned images.
- **interpolation** (*str, optional, default: 'nearest'*) – Method used for interpolation when resize images. Options are OpenCV supported methods.

Returns **res** – Batch of images corresponding to input *batch_ids*.

Return type numpy.ndarray

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

Parameters **id_map** (*dict, optional*) – A dictionary holds mapping from original ids to mapped integer indices of users/items.

Returns **image_modality** – An object of type *ImageModality*.

Return type <cornac.data.ImageModality>

class cornac.data.**GraphModality**(***kwargs*)
Graph modality

Parameters **data** (*List[str], required*) – A list encoding an adjacency matrix, of a user or an item graph, in the sparse triplet format, e.g., data=[('user1', 'user4', 1.0)].

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

Parameters **batch_ids** (*array, required*) – An array containing 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

classmethod from_feature(*features*, *k=5*, *ids=None*, *similarity='cosine'*, *symmetric=False*, *verbose=True*)
Instantiate a GraphModality with a KNN graph build using input features.

Parameters

- **features** (*2d Numpy array, shape: [n_objects, n_features], required*) – A 2d Numpy array of features, e.g., visual, textual, etc.
- **k** (*int, optional, default: 5*) – The number of nearest neighbors
- **ids** (*array, optional, default: None*) – The list of object ids or labels, which align with the rows of features. For instance if you use textual (bag-of-word) features, then "ids" should be the same as the input to cornac.data.TextModality.
- **similarity** (*string, optional, default: "cosine"*) – The similarity measure. At this time only the cosine is supported
- **symmetric** (*bool, optional, default: False*) – When True the resulting KNN-Graph is made symmetric
- **verbose** (*bool, default: False*) – The verbosity flag.

Returns **graph_modality** – GraphModality object.

Return type <cornac.data.GraphModality>

get_node_degree(*in_ids=None*, *out_ids=None*)
Get the "in" and "out" degree for the desired set of nodes

Parameters

- **in_ids** (*array, required*) – An array containing the ids for which to get the "in" degree.
- **out_ids** (*array, required*) – An array containing the ids for which to get the "out" degree.

Returns **Dictionary of the from {node_id**

Return type [in_degree,out_degree]}

get_train_triplet(*train_row_ids*, *train_col_ids*)
Get the subset of relations which align with the training data

Parameters

- **train_row_ids** (*array, required*) – An array containing the ids of training objects (users or items) for which to get the "out" relations.
- **train_col_ids** (*array, required*) – An array containing the ids of training objects (users or items) for whom to get the "in" relations. Please refer to cornac/models/c2pf/recom_c2pf.py for a concrete usage example of this function.

Returns

Return type A subset of the adjacency matrix, in the sparse triplet format, whose elements align with the training set as specified by "train_row_ids" and "train_col_ids".

matrix
Return the adjacency matrix in scipy csr sparse format

class cornac.data.**SentimentModality**(***kwargs*)
Aspect module :param data: A triplet list of user, item, and sentiment information which also a triplet list of aspect, opinion, and sentiment, e.g., data=[('user1', 'item1', [('aspect1', 'opinion1', 'sentiment1')])]. :type data: List[str], required

build(*uid_map=None*, *iid_map=None*, *dok_matrix=None*)
Build the model based on provided list of ordered ids

num_aspects
Return the number of aspects

num_opinions
Return the number of aspects

class cornac.data.**Dataset**(*num_users*, *num_items*, *uid_map*, *iid_map*, *uir_tuple*, *timestamps=None*, *seed=None*)
Training set contains preference matrix

Parameters

- **num_users** (*int, required*) – Number of users.
- **num_items** (*int, required*) – Number of items.
- **uid_map** (OrderDict, required) – The dictionary containing mapping from user original ids to mapped integer indices.
- **iid_map** (OrderDict, required) – The dictionary containing mapping from item original ids to mapped integer indices.
- **uir_tuple** (*tuple, required*) – Tuple of 3 numpy arrays (user_indices, item_indices, rating_values).
- **timestamps** (*numpy.array, optional, default: None*) – Array of timestamps corresponding to observations in *uir_tuple*.
- **seed** (*int, optional, default: None*) – Random seed for reproducing data sampling.

num_users
Number of users in the dataset. For the case of validation or test dataset, this could add up the number of users in the training dataset as well.

Type int

num_items

Number of items in the dataset. For the case of validation or test dataset, this could add up the number of items in the training dataset as well.

Type int

num_ratings
Number of rating observations in the dataset.

Type int

max_rating
Maximum value among the rating observations.

Type float

max_rating
Minimum value among the rating observations.

Type float

global_mean
Average value over the rating observations.

Type float

uir_tuple
Tuple three numpy arrays (user_indices, item_indices, rating_values).

Type tuple

timestamps
Numpy array of timestamps corresponding to feedback in *uir_tuple*. This is only available when input data is in *UIRT* format.

Type numpy.array

user_data
Data organized by user. A dictionary where keys are users, values are tuples of two lists (items, ratings) interacted by the corresponding users.

Type dict

item_data
Data organized by item. A dictionary where keys are items, values are tuples of two lists (users, ratings) interacted with the corresponding items.

Type dict

chrono_user_data
Data organized by user sorted chronologically (timestamps required). A dictionary where keys are users, values are tuples of three chronologically sorted lists (items, ratings, timestamps) interacted by the corresponding users.

Type dict

chrono_item_data
Data organized by item sorted chronologically (timestamps required). A dictionary where keys are items, values are tuples of three chronologically sorted lists (users, ratings, timestamps) interacted with the corresponding items.

Type dict

classmethod build(*data*, *fmt='UIR'*, *global_uid_map=None*, *global_iid_map=None*, *seed=None*, *exclude_unknowns=False*)
Constructing Dataset from given data of specific format.

Parameters

- **data** (*array-like, required*) – Data in the form of triplets (user, item, rating) for UIR format, or quadruplets (user, item, rating, timestamps) for UIRT format.
- **fmt** (*str, default: 'UIR'*) – Format of the input data. Currently, we are supporting:

  'UIR': User, Item, Rating 'UIRT': User, Item, Rating, Timestamp

- **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.
- **seed** (*int, optional, default: None*) – Random seed for reproducing data sampling.
- **exclude_unknowns** (*bool, default: False*) – Ignore unknown users and items.

Returns **res** – Dataset object.

Return type <cornac.data.Dataset>

chrono_item_data
Return data organized by item sorted chronologically (timestamps required). A dictionary where keys are items, values are tuples of three chronologically sorted lists (users, ratings, timestamps) interacted with the corresponding items.

chrono_user_data
Return data organized by user sorted chronologically (timestamps required). A dictionary where keys are users, values are tuples of three chronologically sorted lists (items, ratings, timestamps) interacted by the corresponding users.

csc_matrix
Return the user-item interaction matrix in CSC sparse format

csr_matrix
Return the user-item interaction matrix in CSR sparse format

dok_matrix
Return the user-item interaction matrix in DOK sparse format

classmethod from_uir(*data*, *seed=None*)
Constructing Dataset from UIR (User, Item, Rating) triplet data.

Parameters

- **data** (*array-like, shape: [n_examples, 3]*) – Data in the form of triplets (user, item, rating)
- **seed** (*int, optional, default: None*) – Random seed for reproducing data sampling.

Returns **res** – Dataset object.

Return type <cornac.data.Dataset>

classmethod from_uirt(*data*, *seed=None*)
Constructing Dataset from UIRT (User, Item, Rating, Timestamp) quadruplet data.

Parameters

- **data** (*array-like, shape: [n_examples, 4]*) – Data in the form of triplets (user, item, rating, timestamp)
- **seed** (*int, optional, default: None*) – Random seed for reproducing data sampling.

Returns **res** – Dataset object.

Return type <cornac.data.Dataset>

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)

is_unk_item(*item_idx*)
Return whether or not an item is unknown given the item index

is_unk_user(*user_idx*)
Return whether or not a user is unknown given the user index

item_data
Return data organized by item. A dictionary where keys are items, values are tuples of two lists (users, ratings) interacted with the corresponding items.

item_ids
Return an iterator over the raw item ids

item_indices
Return an iterator over the item indices

item_iter(*batch_size=1*, *shuffle=False*)
Create an iterator over item 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 item indices (array of np.int)

matrix
Return the user-item interaction matrix in CSR sparse format

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

Parameters

- **batch_size** (*int, optional, default = 1*) –
- **shuffle** (*bool, optional, default: False*) – If *True*, orders of triplets will be randomized. If *False*, default orders kept.
- **neg_sampling** (*str, optional, default: 'uniform'*) – How negative item *j* will be sampled. Supported options: {*uniform*, *popularity*}.

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*, *binary=False*, *num_zeros=0*)
Create an iterator over data yielding batch of users, items, and rating values

Parameters

- **batch_size** (*int, optional, default = 1*) –
- **shuffle** (*bool, optional, default: False*) – If *True*, orders of triplets will be randomized. If *False*, default orders kept.
- **binary** (*bool, optional, default: False*) – If *True*, non-zero ratings will be turned into *1*, otherwise, values remain unchanged.
- **num_zeros** (*int, optional, default = 0*) – Number of unobserved ratings (zeros) to be added per user. This could be used for negative sampling. By default, no values are added.

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_data
Return data organized by user. A dictionary where keys are users, values are tuples of two lists (items, ratings) interacted by the corresponding users.

user_ids
Return an iterator over the raw user ids

user_indices
Return an iterator over the user indices

user_iter(*batch_size=1*, *shuffle=False*)
Create an iterator over user 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 user indices (array of np.int)

class cornac.data.**Reader**(*user_set=None*, *item_set=None*, *min_user_freq=1*, *min_item_freq=1*, *bin_threshold=None*, *encoding='utf-8'*, *errors=None*)
Reader class for reading data with different types of format.

Parameters

- **user_set** (*set, default = None*) – Set of users to be selected when reading data. If *None*, all users that appear in the data will be included.
- **item_set** (*set, default = None*) – Set of items to be selected when reading data. If *None*, all items that appear in the data will be included.
- **min_user_freq** (*int, default = 1*) – The minimum frequency of a user to be selected. If *min_user_freq=1*, all users that appear in the data will be included.
- **min_item_freq** (*int, default = 1*) – The minimum frequency of an item to be selected. If *min_item_freq=1*, all items that appear in the data will be included.
- **bin_threshold** (*float, default = None*) – The rating threshold to binarize rating values (turn explicit feedback to implicit feedback). For example, if *bin_threshold = 3.0*, all rating values >= 3.0 will be set to 1.0, and the rest (< 3.0) will be discarded.
- **encoding** (str, default = *utf-8*) – Encoding used to decode the file.

- **errors** (*int, default = None*) – Optional string that specifies how encoding errors are to be handled. Pass 'strict' to raise a ValueError exception if there is an encoding error (None has the same effect), or pass 'ignore' to ignore errors.

read(*fpath*, *fmt='UIR'*, *sep='\t'*, *skip_lines=0*, *id_inline=False*, *parser=None*, ***kwargs*)

Read data and parse line by line based on provided *fmt* or *parser*.

Parameters

- **fpath** (*str*) – Path to the data file.
- **fmt** (str, default: *UIR*) – Line format to be parsed (*UIR* or *UIRT*).
- **sep** (*str, default:*) – The delimiter string.
- **skip_lines** (*int, default: 0*) – Number of first lines to skip
- **id_inline** (*bool, default: False*) – If *True*, user ids corresponding to the line numbers of the file, where all the ids in each line are item ids.
- **parser** (*function, default: None*) – Function takes a list of *str* tokenized by *sep* and returns a list of tuples which will be joined to the final results. If *None*, parser will be determined based on *fmt*.

Returns **tuples** – Data in the form of list of tuples. What inside each tuple depends on *parser* or *fmt*.

Return type list

1.1 Dataset

class cornac.data.dataset.**Dataset**(*num_users*, *num_items*, *uid_map*, *iid_map*, *uir_tuple*, *timestamps=None*, *seed=None*)

Training set contains preference matrix

Parameters

- **num_users** (*int, required*) – Number of users.
- **num_items** (*int, required*) – Number of items.
- **uid_map** (OrderDict, required) – The dictionary containing mapping from user original ids to mapped integer indices.
- **iid_map** (OrderDict, required) – The dictionary containing mapping from item original ids to mapped integer indices.
- **uir_tuple** (*tuple, required*) – Tuple of 3 numpy arrays (user_indices, item_indices, rating_values).
- **timestamps** (*numpy.array, optional, default: None*) – Array of timestamps corresponding to observations in *uir_tuple*.
- **seed** (*int, optional, default: None*) – Random seed for reproducing data sampling.

num_users

Number of users in the dataset. For the case of validation or test dataset, this could add up the number of users in the training dataset as well.

Type int

num_items

Number of items in the dataset. For the case of validation or test dataset, this could add up the number of items in the training dataset as well.

Type int

num_ratings
Number of rating observations in the dataset.

Type int

max_rating
Maximum value among the rating observations.

Type float

max_rating
Minimum value among the rating observations.

Type float

global_mean
Average value over the rating observations.

Type float

uir_tuple
Tuple three numpy arrays (user_indices, item_indices, rating_values).

Type tuple

timestamps
Numpy array of timestamps corresponding to feedback in *uir_tuple*. This is only available when input data is in *UIRT* format.

Type numpy.array

user_data
Data organized by user. A dictionary where keys are users, values are tuples of two lists (items, ratings) interacted by the corresponding users.

Type dict

item_data
Data organized by item. A dictionary where keys are items, values are tuples of two lists (users, ratings) interacted with the corresponding items.

Type dict

chrono_user_data
Data organized by user sorted chronologically (timestamps required). A dictionary where keys are users, values are tuples of three chronologically sorted lists (items, ratings, timestamps) interacted by the corresponding users.

Type dict

chrono_item_data
Data organized by item sorted chronologically (timestamps required). A dictionary where keys are items, values are tuples of three chronologically sorted lists (users, ratings, timestamps) interacted with the corresponding items.

Type dict

classmethod build(*data*, *fmt='UIR'*, *global_uid_map=None*, *global_iid_map=None*, *seed=None*, *exclude_unknowns=False*)

Constructing Dataset from given data of specific format.

Parameters

- **data** (*array-like, required*) – Data in the form of triplets (user, item, rating) for UIR format, or quadruplets (user, item, rating, timestamps) for UIRT format.
- **fmt** (*str, default: 'UIR'*) – Format of the input data. Currently, we are supporting:

  'UIR': User, Item, Rating 'UIRT': User, Item, Rating, Timestamp
- **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.
- **seed** (*int, optional, default: None*) – Random seed for reproducing data sampling.
- **exclude_unknowns** (*bool, default: False*) – Ignore unknown users and items.

Returns **res** – Dataset object.

Return type <cornac.data.Dataset>

chrono_item_data

Return data organized by item sorted chronologically (timestamps required). A dictionary where keys are items, values are tuples of three chronologically sorted lists (users, ratings, timestamps) interacted with the corresponding items.

chrono_user_data

Return data organized by user sorted chronologically (timestamps required). A dictionary where keys are users, values are tuples of three chronologically sorted lists (items, ratings, timestamps) interacted by the corresponding users.

csc_matrix

Return the user-item interaction matrix in CSC sparse format

csr_matrix

Return the user-item interaction matrix in CSR sparse format

dok_matrix

Return the user-item interaction matrix in DOK sparse format

classmethod from_uir(*data*, *seed=None*)

Constructing Dataset from UIR (User, Item, Rating) triplet data.

Parameters

- **data** (*array-like, shape: [n_examples, 3]*) – Data in the form of triplets (user, item, rating)
- **seed** (*int, optional, default: None*) – Random seed for reproducing data sampling.

Returns **res** – Dataset object.

Return type <cornac.data.Dataset>

classmethod from_uirt(*data*, *seed=None*)
Constructing Dataset from UIRT (User, Item, Rating, Timestamp) quadruplet data.

Parameters

- **data** (*array-like, shape: [n_examples, 4]*) – Data in the form of triplets (user, item, rating, timestamp)
- **seed** (*int, optional, default: None*) – Random seed for reproducing data sampling.

Returns **res** – Dataset object.

Return type <cornac.data.Dataset>

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)

is_unk_item(*item_idx*)
Return whether or not an item is unknown given the item index

is_unk_user(*user_idx*)
Return whether or not a user is unknown given the user index

item_data
Return data organized by item. A dictionary where keys are items, values are tuples of two lists (users, ratings) interacted with the corresponding items.

item_ids
Return an iterator over the raw item ids

item_indices
Return an iterator over the item indices

item_iter(*batch_size=1*, *shuffle=False*)
Create an iterator over item 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 item indices (array of np.int)

matrix
Return the user-item interaction matrix in CSR sparse format

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

Parameters

- **batch_size** (*int, optional, default = 1*) –
- **shuffle** (*bool, optional, default: False*) – If *True*, orders of triplets will be randomized. If *False*, default orders kept.
- **neg_sampling** (*str, optional, default: 'uniform'*) – How negative item *j* will be sampled. Supported options: {*uniform*, *popularity*}.

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*, *binary=False*, *num_zeros=0*)
Create an iterator over data yielding batch of users, items, and rating values

Parameters

- **batch_size** (*int, optional, default = 1*) –
- **shuffle** (*bool, optional, default: False*) – If *True*, orders of triplets will be randomized. If *False*, default orders kept.
- **binary** (*bool, optional, default: False*) – If *True*, non-zero ratings will be turned into *1*, otherwise, values remain unchanged.
- **num_zeros** (*int, optional, default = 0*) – Number of unobserved ratings (zeros) to be added per user. This could be used for negative sampling. By default, no values are added.

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_data
Return data organized by user. A dictionary where keys are users, values are tuples of two lists (items, ratings) interacted by the corresponding users.

user_ids
Return an iterator over the raw user ids

user_indices
Return an iterator over the user indices

user_iter(*batch_size=1*, *shuffle=False*)
Create an iterator over user 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 user indices (array of np.int)

1.2 Modality

class cornac.data.modality.**FeatureModality**(*features=None*, *ids=None*, *normalized=False*, ***kwargs*)
Modality that contains features in general

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.modality.**Modality**(***kwargs*)
Generic class of Modality to extend from

cornac.data.modality.**fallback_feature**(*func*)
Decorator to fallback to *batch_feature* in FeatureModality

1.3 Graph Modality

class cornac.data.graph.**GraphModality**(***kwargs*)
Graph modality

Parameters **data** (*List[str], required*) – A list encoding an adjacency matrix, of a user or an item graph, in the sparse triplet format, e.g., data=[('user1', 'user4', 1.0)].

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

Parameters **batch_ids** (*array, required*) – An array containing 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

classmethod from_feature(*features*, *k=5*, *ids=None*, *similarity='cosine'*, *symmetric=False*, *verbose=True*)
Instantiate a GraphModality with a KNN graph build using input features.

Parameters

- **features** (*2d Numpy array, shape: [n_objects, n_features], required*) – A 2d Numpy array of features, e.g., visual, textual, etc.
- **k** (*int, optional, default: 5*) – The number of nearest neighbors
- **ids** (*array, optional, default: None*) – The list of object ids or labels, which align with the rows of features. For instance if you use textual (bag-of-word) features, then "ids" should be the same as the input to cornac.data.TextModality.
- **similarity** (*string, optional, default: "cosine"*) – The similarity measure. At this time only the cosine is supported

- **symmetric** (*bool, optional, default: False*) – When True the resulting KNN-Graph is made symmetric
- **verbose** (*bool, default: False*) – The verbosity flag.

Returns **graph_modality** – GraphModality object.

Return type <cornac.data.GraphModality>

get_node_degree(*in_ids=None*, *out_ids=None*)
Get the "in" and "out" degree for the desired set of nodes

Parameters

- **in_ids** (*array, required*) – An array containing the ids for which to get the "in" degree.
- **out_ids** (*array, required*) – An array containing the ids for which to get the "out" degree.

Returns **Dictionary of the from {node_id**

Return type [in_degree,out_degree]}

get_train_triplet(*train_row_ids*, *train_col_ids*)
Get the subset of relations which align with the training data

Parameters

- **train_row_ids** (*array, required*) – An array containing the ids of training objects (users or items) for which to get the "out" relations.
- **train_col_ids** (*array, required*) – An array containing the ids of training objects (users or items) for whom to get the "in" relations. Please refer to cornac/models/c2pf/recom_c2pf.py for a concrete usage example of this function.

Returns

Return type A subset of the adjacency matrix, in the sparse triplet format, whose elements align with the training set as specified by "train_row_ids" and "train_col_ids".

matrix
Return the adjacency matrix in scipy csr sparse format

1.4 Text Modality

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.

Parameters **texts** (*List[str], required*) – Input list of texts to be tokenized.

Returns **tokens**

Return type List[List[str]]

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

Parameters **t** (*str, required*) – Input text to be tokenized.

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.

Parameters

- **sep** (*str, optional, default:* ' ') – Separator string used to split text into tokens.
- **pre_rules** (*List[Callable[[str], str]], optional*) – List of callable lambda functions to apply on text before tokenization.
- **stop_words** (*Union[List, str], optional*) – List of stop-words to be ignored during tokenization, or key of built-in stop-word lists (e.g., english).

batch_tokenize(*texts: List[str]*) → List[List[str]]

Splitting a corpus with multiple text documents.

Parameters **texts** (*List[str], required*) – Input list of texts to be tokenized.

Returns tokens

Return type List[List[str]]

tokenize(*t: str*) → List[str]

Splitting text into tokens.

Parameters **t** (*str, required*) – Input text to be tokenized.

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.

Parameters

- **idx2tok** (*List[str], required*) – List of tokens where list indices are corresponding to their mapped integer indices.
- **use_special_tokens** (*bool, optional, default: False*) – If *True*, vocabulary will include *SPECIAL_TOKENS*.

build_tok2idx()

Build a mapping between tokens to their integer indices

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).

Parameters

- **sequences** (*List[List[str]], required*) – Corpus of multiple lists of string tokens.
- **max_vocab** (*int, optional*) – Limit for size of the vocabulary. If specified, tokens will be ranked based on counts and gathered top-down until reach *max_vocab*.
- **min_freq** (*int, optional, default: 1*) – Cut-off threshold for tokens based on their counts.

- **use_special_tokens** (*bool, optional, default: False*) – If *True*, vocabulary will include *SPECIAL_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.

Parameters

- **tokens** (*List[str], required*) – List of string tokens.
- **max_vocab** (*int, optional*) – Limit for size of the vocabulary. If specified, tokens will be ranked based on counts and gathered top-down until reach *max_vocab*.
- **min_freq** (*int, optional, default: 1*) – Cut-off threshold for tokens based on their counts.
- **use_special_tokens** (*bool, optional, default: False*) – If *True*, vocabulary will include *SPECIAL_TOKENS*.

classmethod load(*path*)

Load a vocabulary from *path* to a pickle file.

save(*path*)

Save idx2tok into a pickle file.

Parameters **path** (*str, required*) – Path to store the dictionary on disk.

to_idx(*tokens: List[str]*) → List[int]

Convert a list of *tokens* to their integer indices.

Parameters **tokens** (*List[str], required*) – List of string tokens.

Returns **indices** – List of integer indices corresponding to input *tokens*.

Return type List[int]

to_text(*indices: List[int], sep=' '*) → List[str]

Convert a list of integer *indices* to their tokens.

Parameters

- **indices** (*List[int], required*) – List of token integer indices.
- **sep** (*str, optional, default: ' '*) – Separator string used to connect tokens.

Returns **text** – Aggregated text of tokens seperated by *sep*.

Return type str

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_doc_freq: int = 1, max_features: int = 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** (*float in range [0.0, 1.0] or int, default=1.0*) – When building the vocabulary ignore terms that have a document frequency strictly higher than the given threshold (corpus-specific stop words). If float, the value represents a proportion of documents, int for absolute counts. If *vocab* is not None, this will be ignored.
- **min_doc_freq** (*float in range [0.0, 1.0] or int, default=1*) – When building the vocabulary ignore terms that have a document frequency strictly lower than the given threshold. This value is also called cut-off in the literature. If float, the value represents a proportion of documents, int absolute counts. If *vocab* is not None, this will be ignored.
- **max_features** (*int or None, optional, 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.
- **binary** (*boolean, default=False*) – If True, all non zero counts are set to 1.
- **Reference** –
- **---------** –
- **https** (*//github.com/scikit-learn/scikit-learn/blob/master/sklearn/feature_extraction/text.py#L790*) –

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 **count_vectorizer** – An object of type *CountVectorizer*.

Return type <cornac.data.text.CountVectorizer>

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.**TextModality**(*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_doc_freq: int = 1*, *tfidf_params: Dict = None*, ***kwargs*)

Text modality

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** (*float in range [0.0, 1.0] or int, default=1.0*) – When building the vocabulary ignore terms that have a document frequency strictly higher than the given threshold (corpus-specific stop words). If float, the value represents a proportion of documents, int for absolute counts. If *vocab* is not None, this will be ignored.
- **min_doc_freq** (*float in range [0.0, 1.0] or int, default=1*) – When building the vocabulary ignore terms that have a document frequency strictly lower than the given threshold. This value is also called cut-off in the literature. If float, the value represents a proportion of documents, int absolute counts. If *vocab* is not None, this will be ignored.
- **tfidf_params** (*dict or None, optional, default=None*) – If *None*, a default arguments of <cornac.data.text.IfidfVectorizer> will be used. List of parameters:

  'binary' [boolean, default=False] If True, all non zero counts are set to 1.

  'norm' ['l1', 'l2' or None, optional, default='l2'] Each output row will have unit norm, either: * 'l2': Sum of squares of vector elements is 1. The cosine similarity between two vectors is their dot product when l2 norm has been applied. * 'l1': Sum of absolute values of vector elements is 1. See utils.common.normalize()

  'use_idf' [boolean, default=True] Enable inverse-document-frequency reweighting.

  'smooth_idf' [boolean, default=True] Smooth idf weights by adding one to document frequencies, as if an extra document was seen containing every term in the collection exactly once. Prevents zero divisions.

  'sublinear_tf' [boolean (default=False)] Apply sublinear tf scaling, i.e. replace tf with 1 + log(tf).

batch_seq(*batch_ids*, *max_length=None*)

Return a numpy matrix of text sequences containing token ids with size=(len(batch_ids), max_length).

Parameters

- **batch_ids** (*Union[List, numpy.array], required*) – An array containing the ids of rows of text sequences to be returned.

- **max_length** (*int, optional*) – Cut-off length of returned sequences. If *None*, it will be inferred based on retrieved sequences.

Returns **batch_sequences** – Batch of sequences with zero-padding at the end.

Return type numpy.ndarray

batch_tfidf(*batch_ids*, *keep_sparse=False*)
Return matrix of TF-IDF features corresponding to provided batch_ids

Parameters

- **batch_ids** (*array*) – An array of ids to retrieve the corresponding features.
- **keep_sparse** (*bool, default = False*) – If *True*, the return feature matrix will be a *scipy.sparse.csr_matrix*. Otherwise, it will be a dense matrix.

Returns **batch_tfidf** – Batch of TF-IDF representations corresponding to input *batch_ids*.

Return type numpy.ndarray

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

Parameters **id_map** (*dict, optional*) – A dictionary holds mapping from original ids to mapped integer indices of users/items.

Returns **text_modality** – An object of type *TextModality*.

Return type <cornac.data.TextModality>

tfidf_matrix
Return tf-idf matrix.

1.5 Image Modality

class cornac.data.image.**ImageModality**(***kwargs*)
Image modality

Parameters

- **images** (*Union[List, numpy.ndarray], optional*) – A list or tensor of images that the row indices are aligned with user/item in *ids*.
- **paths** (*List[str], optional*) – A list of paths, to images stored on disk, which the row indices are aligned with user/item in *ids*..

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

Parameters

- **batch_ids** (*Union[List, numpy.array], required*) – An array containing the ids of rows of images to be returned.
- **target_size** (*tuple, optional, default: (256, 256)*) – Size (width, height) of returned images to be resized.
- **color_mode** (*str, optional, default: 'rgb'*) – Color mode of returned images.
- **interpolation** (*str, optional, default: 'nearest'*) – Method used for interpolation when resize images. Options are OpenCV supported methods.

Returns **res** – Batch of images corresponding to input *batch_ids*.

Return type numpy.ndarray

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

Parameters **id_map** (*dict, optional*) – A dictionary holds mapping from original ids to mapped integer indices of users/items.

Returns **image_modality** – An object of type *ImageModality*.

Return type <cornac.data.ImageModality>

1.6 Sentiment Modality

class cornac.data.sentiment.**SentimentModality**(***kwargs*)
Aspect module :param data: A triplet list of user, item, and sentiment information which also a triplet list of aspect, opinion, and sentiment, e.g., data=[('user1', 'item1', [('aspect1', 'opinion1', 'sentiment1')])]. :type data: List[str], required

build(*uid_map=None*, *iid_map=None*, *dok_matrix=None*)
Build the model based on provided list of ordered ids

num_aspects
Return the number of aspects

num_opinions
Return the number of aspects

1.7 Reader

class cornac.data.reader.**Reader**(*user_set=None*, *item_set=None*, *min_user_freq=1*, *min_item_freq=1*, *bin_threshold=None*, *encoding='utf-8'*, *errors=None*)
Reader class for reading data with different types of format.

Parameters

- **user_set** (*set, default = None*) – Set of users to be selected when reading data. If *None*, all users that appear in the data will be included.
- **item_set** (*set, default = None*) – Set of items to be selected when reading data. If *None*, all items that appear in the data will be included.
- **min_user_freq** (*int, default = 1*) – The minimum frequency of a user to be selected. If *min_user_freq=1*, all users that appear in the data will be included.
- **min_item_freq** (*int, default = 1*) – The minimum frequency of an item to be selected. If *min_item_freq=1*, all items that appear in the data will be included.
- **bin_threshold** (*float, default = None*) – The rating threshold to binarize rating values (turn explicit feedback to implicit feedback). For example, if *bin_threshold = 3.0*, all rating values >= 3.0 will be set to 1.0, and the rest (< 3.0) will be discarded.
- **encoding** (str, default = *utf-8*) – Encoding used to decode the file.

- **errors** (*int, default = None*) – Optional string that specifies how encoding errors are to be handled. Pass 'strict' to raise a ValueError exception if there is an encoding error (None has the same effect), or pass 'ignore' to ignore errors.

read(*fpath*, *fmt='UIR'*, *sep='\t'*, *skip_lines=0*, *id_inline=False*, *parser=None*, ***kwargs*)

Read data and parse line by line based on provided *fmt* or *parser*.

Parameters

- **fpath** (*str*) – Path to the data file.
- **fmt** (str, default: *UIR*) – Line format to be parsed (*UIR* or *UIRT*).
- **sep** (*str, default:*) – The delimiter string.
- **skip_lines** (*int, default: 0*) – Number of first lines to skip
- **id_inline** (*bool, default: False*) – If *True*, user ids corresponding to the line numbers of the file, where all the ids in each line are item ids.
- **parser** (*function, default: None*) – Function takes a list of *str* tokenized by *sep* and returns a list of tuples which will be joined to the final results. If *None*, parser will be determined based on *fmt*.

Returns **tuples** – Data in the form of list of tuples. What inside each tuple depends on *parser* or *fmt*.

Return type list

cornac.data.reader.**read_text**(*fpath*, *sep=None*, *encoding='utf-8'*, *errors=None*)

Read text file and return two lists of text documents and corresponding ids. If *sep* is None, only return one list containing elements are lines of text in the original file.

Parameters

- **fpath** (*str*) – Path to the data file
- **sep** (*str, default = None*) – The delimiter string used to split *id* and *text*. Each line is assumed containing an *id* followed by corresponding *text* document. If *None*, each line will be a *str* in returned list.
- **encoding** (str, default = *utf-8*) – Encoding used to decode the file.
- **errors** (*int, default = None*) – Optional string that specifies how encoding errors are to be handled. Pass 'strict' to raise a ValueError exception if there is an encoding error (None has the same effect), or pass 'ignore' to ignore errors.

CHAPTER 2

Models

2.1 Recommender (Generic Class)

class cornac.models.recommender.**Recommender**(*name*, *trainable=True*, *verbose=False*)

Generic class for a recommender model. All recommendation models should inherit from this class

Parameters

- **name** (*str, required*) – The name of the recommender model
- **trainable** (*boolean, optional, default: True*) – When False, the model is not trainable

default_score()

Overwrite this function if your algorithm has special treatment for cold-start problem

early_stop(*min_delta=0.0*, *patience=0*)

Check if training should be stopped when validation loss has stopped improving.

Parameters

- **min_delta** (*float, optional, default: 0.*) – The minimum increase in monitored value on validation set to be considered as improvement, i.e. an increment of less than *min_delta* will count as no improvement.
- **patience** (*int, optional, default: 0*) – Number of epochs with no improvement after which training should be stopped.

Returns **res** – Return *True* if model training should be stopped (no improvement on validation set), otherwise return *False*.

Return type bool

fit(*train_set*, *val_set=None*)

Fit the model to observations. Need to

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.
- **val_set** (*object of type TestSet, optional, default: None*) – An object containing the user-item preference for model selection purposes (e.g., early stopping). Please refer to the class TestSet in the "data" module for details.

Returns self

Return type object

monitor_value()
Calculating monitored value used for early stopping on validation set (*val_set*). This function will be called by *early_stop()* function. Note: *val_set* could be *None* thus it needs to be checked before usage.

Returns

Return type raise NotImplementedError

rank(*user_idx*, *item_indices=None*)
Rank all test items for a given user.

Parameters

- **user_idx** (*int, required*) – The index of the user for whom to perform item raking.
- **item_indices** (*1d array, optional, default: None*) – A list of candidate item indices to be ranked by the user. If *None*, list of ranked known item indices and their scores will be returned

Returns

- Tuple of *item_rank*, and *item_scores*. The order of values
- *in item_scores are corresponding to the order of their ids in item_ids*

rate(*user_idx*, *item_idx*, *clipping=True*)
Give a rating score between pair of user and item

Parameters

- **user_idx** (*int, required*) – The index of the user for whom to perform item raking.
- **item_idx** (*int, required*) – The index of the item to be rated by the user.
- **clipping** (*bool, default: True*) – Whether to clip the predicted rating value.

Returns A rating score of the user for the item

Return type A scalar

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

Parameters

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

2.2 Collaborative Context Poisson Factorization (C2PF)

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 modality 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*, *val_set=None*)

Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

score(*user_idx*, *item_idx=None*)

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

Parameters

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

2.3 Multi-Task Explainable Recommendation (MTER)

class cornac.models.mter.recom_mter.**MTER**(*name='MTER'*, *rating_scale=5.0*, *n_user_factors=15*, *n_item_factors=15*, *n_aspect_factors=12*, *n_opinion_factors=12*, *n_bpr_samples=1000*, *n_element_samples=50*, *lambda_reg=0.1*, *lambda_bpr=10*, *n_epochs=200000*, *lr=0.1*, *n_threads=0*, *trainable=True*, *verbose=False*, *init_params={}*, *seed=None*)

Multi-Task Explainable Recommendation

Parameters

- **name** (*string, optional, default: 'MTER'*) – The name of the recommender model.
- **rating_scale** (*float, optional, default: 5.0*) – The maximum rating score of the dataset.
- **n_user_factors** (*int, optional, default: 15*) – The dimension of the user latent factors.
- **n_item_factors** (*int, optional, default: 15*) – The dimension of the item latent factors.
- **n_aspect_factors** (*int, optional, default: 12*) – The dimension of the aspect latent factors.
- **n_opinion_factors** (*int, optional, default: 12*) – The dimension of the opinion latent factors.
- **n_bpr_samples** (*int, optional, default: 1000*) – The number of samples from all BPR pairs.

- **n_element_samples** (*int, optional, default: 50*) – The number of samples from all ratings in each iteration.
- **lambda_reg** (*float, optional, default: 0.1*) – The regularization parameter.
- **lambda_bpr** (*float, optional, default: 10.0*) – The regularization parameter for BPR.
- **n_epochs** (*int, optional, default: 200000*) – Maximum number of epochs for training.
- **lr** (*float, optional, default: 0.1*) – The learning rate for optimization
- **n_threads** (*int, optional, default: 0*) – Number of parallel threads for training. If n_threads=0, all CPU cores will be utilized. If seed is not None, n_threads=1 to remove randomness from parallelization.
- **trainable** (*boolean, optional, default: True*) – When False, the model is not trained and Cornac assumes that the model already pre-trained (U, I, A, O, G1, G2, and G3 are not None).
- **verbose** (*boolean, optional, default: False*) – When True, running logs are displayed.
- **init_params** (*dictionary, optional, default: {}*) – List of initial parameters, e.g., init_params = {'U':U, 'I':I, 'A':A, 'O':O, 'G1':G1, 'G2':G2, 'G3':G3}

  U: ndarray, shape (n_users, n_user_factors) The user latent factors, optional initialization via init_params

  I: ndarray, shape (n_items, n_item_factors) The item latent factors, optional initialization via init_params

  A: ndarray, shape (num_aspects+1, n_aspect_factors) The aspect latent factors, optional initialization via init_params

  O: ndarray, shape (num_opinions, n_opinion_factors) The opinion latent factors, optional initialization via init_params

  G1: ndarray, shape (n_user_factors, n_item_factors, n_aspect_factors) The core tensor for user, item, and aspect factors, optional initialization via init_params

  G2: ndarray, shape (n_user_factors, n_aspect_factors, n_opinion_factors) The core tensor for user, aspect, and opinion factors, optional initialization via init_params

  G3: ndarray, shape (n_item_factors, n_aspect_factors, n_opinion_factors) The core tensor for item, aspect, and opinion factors, optional initialization via init_params
- **seed** (*int, optional, default: None*) – Random seed for parameters initialization.

References

Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. 2018. Explainable Recommendation via Multi-Task Learning in Opinionated Text Data. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18). ACM, New York, NY, USA, 165-174. DOI: https://doi.org/10.1145/3209978.3210010

fit(*train_set*, *val_set=None*)

Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

score(*user_idx*, *item_idx=None*)

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

Parameters

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

2.4 Probabilistic Collaborative Representation Learning (PCRL)

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 modality 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*, *val_set=None*)

Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

score (*user_idx*, *item_idx=None*)

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

Parameters

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

2.5 VAE for Collaborative Filtering (VAECF)

class cornac.models.vaecf.recom_vaecf.**VAECF**(*name='VAECF', k=10, h=20, n_epochs=100, batch_size=100, learning_rate=0.001, beta=1.0, trainable=True, verbose=False, seed=None, use_gpu=False*)

Variational Autoencoder for Collaborative Filtering.

Parameters

- **k** (*int, optional, default: 10*) – The dimension of the stochastic user factors "z".
- **h** (*int, optional, default: 20*) – The dimension of the deterministic hidden layer.
- **n_epochs** (*int, optional, default: 100*) – The number of epochs for SGD.
- **batch_size** (*int, optional, default: 100*) – The batch size.
- **learning_rate** (*float, optional, default: 0.001*) – The learning rate for SGD_RMSProp.
- **beta** (*float, optional, default: 1.*) – The weight of the KL term as in beta-VAE.
- **name** (*string, optional, default: 'VAECF'*) – 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.
- **verbose** (*boolean, optional, default: False*) – When True, some running logs are displayed.
- **seed** (*int, optional, default: None*) – Random seed for parameters initialization.
- **use_gpu** (*boolean, optional, default: False*) – If True and your system supports CUDA then training is performed on GPUs.

References

- Liang, Dawen, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. "Variational autoencoders for collaborative filtering." In Proceedings of the 2018 World Wide Web Conference on World Wide Web, pp. 689-698.

fit(*train_set, val_set=None*)

Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

score(*user_idx*, *item_idx=None*)

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

Parameters

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

2.6 Collaborative Variational Autoencoder (CVAE)

class cornac.models.cvae.recom_cvae.**CVAE**(*name='CVAE', z_dim=50, n_epochs=100, lambda_u=0.0001, lambda_v=0.001, lambda_r=10, lambda_w=0.0001, lr=0.001, a=1, b=0.01, input_dim=8000, vae_layers=[200, 100], act_fn='sigmoid', loss_type='cross-entropy', batch_size=128, init_params=None, trainable=True, seed=None, verbose=True*)

Collaborative Variational Autoencoder

Parameters

- **z_dim** (*int, optional, default: 50*) – The dimension of the user and item latent factors.
- **n_epochs** (*int, optional, default: 100*) – Maximum number of epochs for training.
- **lambda_u** (*float, optional, default: 1e-4*) – The regularization hyper-parameter for user latent factor.
- **lambda_v** (*float, optional, default: 0.001*) – The regularization hyper-parameter for item latent factor.
- **lambda_r** (*float, optional, default: 10.0*) – Parameter that balance the focus on content or ratings
- **lambda_w** (*float, optional, default: 1e-4*) – The regularization for VAE weights
- **lr** (*float, optional, default: 0.001*) – Learning rate in the auto-encoder training
- **a** (*float, optional, default: 1*) – The confidence of observed ratings.
- **b** (*float, optional, default: 0.01*) – The confidence of unseen ratings.
- **input_dim** (*int, optional, default: 8000*) – The size of input vector
- **vae_layers** (*list, optional, default: [200, 100]*) – The list containing size of each layers in neural network structure

- **act_fn** (*str, default: 'sigmoid'*) – Name of the activation function used for the variational auto-encoder. Supported functions: [‘sigmoid’, ‘tanh’, ‘elu’, ‘relu’, ‘relu6’, ‘leaky_relu’, ‘identity’]
- **loss_type** (*String, optional, default: "cross-entropy"*) – Either “cross-entropy” or “rmse” The type of loss function in the last layer
- **batch_size** (*int, optional, default: 128*) – The batch size for SGD.
- **init_params** (*dict, optional, default: {'U':None, 'V':None}*) – Initial U and V latent matrix
- **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

Collaborative Variational Autoencoder for Recommender Systems X. Li and J. She ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2017

http://eelxpeng.github.io/assets/paper/Collaborative_Variational_Autoencoder.pdf

fit(*train_set*, *val_set=None*)

Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns **self**

Return type object

score(*user_idx*, *item_idx=None*)

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

Parameters

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

2.7 Generalized Matrix Factorization (NeuMF)

class cornac.models.ncf.recom_gmf.**GMF**(*name='GMF'*, *num_factors=8*, *regs=(0.0, 0.0)*, *num_epochs=20*, *batch_size=256*, *num_neg=4*, *lr=0.001*, *learner='adam'*, *early_stopping=None*, *trainable=True*, *verbose=True*, *seed=None*)

Generalized Matrix Factorization.

Parameters

- **num_factors** (*int, optional, default: 8*) – Embedding size of MF model.
- **regs** (*float, optional, default: 0.*) – Regularization for user and item embeddings.
- **num_epochs** (*int, optional, default: 20*) – Number of epochs.
- **batch_size** (*int, optional, default: 256*) – Batch size.
- **num_neg** (*int, optional, default: 4*) – Number of negative instances to pair with a positive instance.
- **lr** (*float, optional, default: 0.001*) – Learning rate.
- **learner** (*str, optional, default: 'adam'*) – Specify an optimizer: adagrad, adam, rmsprop, sgd
- **early_stopping** (*{min_delta: float, patience: int}, optional, default: None*) –

  If *None*, no early stopping. Meaning of the arguments: - *min_delta*: the minimum increase in monitored valu

  - *patience*: number of epochs with no improvement after which training should be stopped.
- **name** (*string, optional, default: 'GMF'*) – 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.
- **verbose** (*boolean, optional, default: False*) – When True, some running logs are displayed.
- **seed** (*int, optional, default: None*) – Random seed for parameters initialization.

References

- He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017, April). Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web (pp. 173-182).

fit(*train_set*, *val_set=None*)

Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

monitor_value()

Calculating monitored value used for early stopping on validation set (*val_set*). This function will be called by *early_stop()* function.

Returns **res** – Monitored value on validation set. Return *None* if *val_set* is *None*.

Return type float

score(*user_idx*, *item_idx=None*)

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

Parameters

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

2.8 Indexable Bayesian Personalized Ranking (IBPR)

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*, *val_set=None*)
Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

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

Parameters

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

2.9 Matrix Co-Factorization (MCF)

class cornac.models.mcf.recom_mcf.**MCF**(*k=5*, *max_iter=100*, *learning_rate=0.001*, *gamma=0.9*, *lamda=0.001*, *name='MCF'*, *trainable=True*, *verbose=False*, *init_params={}*, *seed=None*)
Matrix Co-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: 'MCF'*) – 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 (U and V are not None).
- **network** (*item-affinity*) –
- **verbose** (*boolean, optional, default: False*) – When True, some running logs are displayed.
- **init_params** (*dictionary, optional, default: {}*) – 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. Z: a csc_matrix of shape (n_items,k), containing the "Also-Viewed" item latent factors.
- **seed** (*int, optional, default: None*) – Random seed for parameters initialization.

References

- Park, Chanyoung, Donghyun Kim, Jinoh Oh, and Hwanjo Yu. "Do Also-Viewed Products Help User Rating Prediction?." In Proceedings of WWW, pp. 1113-1122. 2017.

fit (*train_set*, *val_set=None*)

Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

score (*user_idx*, *item_idx=None*)

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

Parameters

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

2.10 Multi-Layer Perceptron (MLP)

class cornac.models.ncf.recom_mlp.**MLP**(*name='MLP'*, *layers=(64, 32, 16, 8)*, *act_fn='relu'*, *reg_layers=(0.0, 0.0, 0.0, 0.0)*, *num_epochs=20*, *batch_size=256*, *num_neg=4*, *lr=0.001*, *learner='adam'*, *early_stopping=None*, *trainable=True*, *verbose=True*, *seed=None*)

Multi-Layer Perceptron.

Parameters

- **layers** (*list, optional, default: [64, 32, 16, 8]*) – MLP layers. Note that the first layer is the concatenation of user and item embeddings. So layers[0]/2 is the embedding size.
- **act_fn** (*str, default: 'relu'*) – Name of the activation function used for the MLP layers. Supported functions: ['sigmoid', 'tanh', 'elu', 'relu', 'selu, 'relu6', 'leaky_relu']
- **reg_layers** (*list, optional, default: [0., 0., 0., 0.]*) – Regularization for each MLP layer, reg_layers[0] is the regularization for embeddings.
- **num_epochs** (*int, optional, default: 20*) – Number of epochs.
- **batch_size** (*int, optional, default: 256*) – Batch size.
- **num_neg** (*int, optional, default: 4*) – Number of negative instances to pair with a positive instance.
- **lr** (*float, optional, default: 0.001*) – Learning rate.
- **learner** (*str, optional, default: 'adam'*) – Specify an optimizer: adagrad, adam, rmsprop, sgd
- **early_stopping** (*{min_delta: float, patience: int}, optional, default: None*) –

  If *None*, no early stopping. Meaning of the arguments: - *min_delta*: the minimum increase in monitored valu

  - *patience*: number of epochs with no improvement after which training should be stopped.
- **name** (*string, optional, default: 'MLP'*) – 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.
- **verbose** (*boolean, optional, default: False*) – When True, some running logs are displayed.
- **seed** (*int, optional, default: None*) – Random seed for parameters initialization.

References

- He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017, April). Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web (pp. 173-182).

fit(*train_set*, *val_set=None*)
Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

monitor_value()
Calculating monitored value used for early stopping on validation set (*val_set*). This function will be called by *early_stop()* function.

Returns **res** – Monitored value on validation set. Return *None* if *val_set* is *None*.

Return type float

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

Parameters

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

2.11 Neural Matrix Factorization (NeuMF)

class cornac.models.ncf.recom_neumf.**NeuMF**(*name='NeuMF'*, *num_factors=8*, *layers=(64, 32, 16, 8)*, *act_fn='relu'*, *reg_mf=0.0*, *reg_layers=(0.0, 0.0, 0.0, 0.0)*, *num_epochs=20*, *batch_size=256*, *num_neg=4*, *lr=0.001*, *learner='adam'*, *early_stopping=None*, *trainable=True*, *verbose=True*, *seed=None*)
Neural Matrix Factorization.

Parameters

- **num_factors** (*int, optional, default: 8*) – Embedding size of MF model.
- **layers** (*list, optional, default: [64, 32, 16, 8]*) – MLP layers. Note that the first layer is the concatenation of user and item embeddings. So layers[0]/2 is the embedding size.
- **act_fn** (*str, default: 'relu'*) – Name of the activation function used for the MLP layers. Supported functions: [‘sigmoid’, ‘tanh’, ‘elu’, ‘relu’, ‘selu, ‘relu6’, ‘leaky_relu’]
- **reg_mf** (*float, optional, default: 0.*) – Regularization for MF embeddings.

- **reg_layers** (*list, optional, default: [0., 0., 0., 0.]*) – Regularization for each MLP layer, reg_layers[0] is the regularization for embeddings.
- **num_epochs** (*int, optional, default: 20*) – Number of epochs.
- **batch_size** (*int, optional, default: 256*) – Batch size.
- **num_neg** (*int, optional, default: 4*) – Number of negative instances to pair with a positive instance.
- **lr** (*float, optional, default: 0.001*) – Learning rate.
- **learner** (*str, optional, default: 'adam'*) – Specify an optimizer: adagrad, adam, rmsprop, sgd
- **early_stopping** (*{min_delta: float, patience: int}, optional, default: None*) –

  If *None*, no early stopping. Meaning of the arguments: - *min_delta*: the minimum increase in monitored valu

  - *patience*: number of epochs with no improvement after which training should be stopped.
- **name** (*string, optional, default: 'NeuMF'*) – 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.
- **verbose** (*boolean, optional, default: False*) – When True, some running logs are displayed.
- **seed** (*int, optional, default: None*) – Random seed for parameters initialization.

References

- He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017, April). Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web (pp. 173-182).

fit(*train_set*, *val_set=None*)

Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

monitor_value()

Calculating monitored value used for early stopping on validation set (*val_set*). This function will be called by *early_stop()* function.

Returns **res** – Monitored value on validation set. Return *None* if *val_set* is *None*.

Return type float

pretrain(*gmf_model*, *mlp_model*, *alpha=0.5*)
Provide pre-trained GMF and MLP models. Section 3.4.1 of the paper.

Parameters

- **gmf_model** (*object of type GMF, required*) – Reference to trained/fitted GMF model.
- **gmf_model** – Reference to trained/fitted GMF model.
- **alpha** (*float, optional, default: 0.5*) – Hyper-parameter determining the trade-off between the two pre-trained models. Details are described in the section 3.4.1 of the paper.

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

Parameters

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

2.12 Online Indexable Bayesian Personalized Ranking (OIBPR)

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*, *val_set=None*)

Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

score (*user_idx*, *item_idx=None*)

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

Parameters

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

2.13 Visual Matrix Factorization (VMF)

class cornac.models.vmf.recom_vmf.**VMF**(*name='VMF', k=10, d=10, n_epochs=100, batch_size=100, learning_rate=0.001, gamma=0.9, lambda_u=0.001, lambda_v=0.001, lambda_p=1.0, lambda_e=10.0, trainable=True, verbose=False, use_gpu=False, init_params={}, seed=None*)

Visual Matrix Factorization.

Parameters

- **k** (*int, optional, default: 10*) – The dimension of the user and item factors.
- **d** (*int, optional, default: 10*) – The dimension of the user visual factors.
- **n_epochs** (*int, optional, default: 100*) – 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.
- **lambda_u** (*float, optional, default: 0.001*) – The regularization parameter for user factors.
- **lambda_v** (*float, optional, default: 0.001*) – The regularization parameter for item factors.
- **lambda_p** (*float, optional, default: 1.0*) – The regularization parameter for user visual factors.
- **lambda_e** (*float, optional, default: 10.*) – The regularization parameter for the kernel embedding matrix
- **lambda_u** – The regularization parameter for user factors.
- **name** (*string, optional, default: 'VMF'*) – 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 (The parameters of the model U, V, P, E are not None).
- **visual_features** (*See "cornac/examples/vmf_example.py" for an example of how to use cornac's visual modality to load and provide the "item visual features" for VMF.*) –
- **verbose** (*boolean, optional, default: False*) – When True, some running logs are displayed.
- **init_params** (*dictionary, optional, default: {}*) – List of initial parameters, e.g., init_params = {'U':U, 'V':V, 'P': P, 'E': E}.

  U: numpy array of shape (n_users,k), user latent factors. V: numpy array of shape (n_items,k), item latent factors. P: numpy array of shape (n_users,d), user visual latent factors. E: numpy array of shape (d,c), embedding kernel matrix.
- **seed** (*int, optional, default: None*) – Random seed for parameters initialization.

References

- Park, Chanyoung, Donghyun Kim, Jinoh Oh, and Hwanjo Yu. "Do Also-Viewed Products Help User Rating Prediction?." In Proceedings of WWW, pp. 1113-1122. 2017.

fit(*train_set*, *val_set=None*)
Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

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

Parameters

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

2.14 Collaborative Deep Ranking (CDR)

class cornac.models.cdr.recom_cdr.**CDR**(*name='CDR'*, *k=50*, *autoencoder_structure=None*, *act_fn='relu'*, *lambda_u=0.1*, *lambda_v=100*, *lambda_w=0.1*, *lambda_n=1000*, *corruption_rate=0.3*, *learning_rate=0.001*, *dropout_rate=0.1*, *batch_size=128*, *max_iter=100*, *trainable=True*, *verbose=True*, *vocab_size=8000*, *init_params=None*, *seed=None*)
Collaborative Deep Ranking.

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.
- **autoencoder_structure** (*list, default: None*) – The number of neurons of encoder/decoder layer for SDAE. For example, autoencoder_structure = [200], the SDAE structure will be [vocab_size, 200, k, 200, vocab_size]
- **act_fn** (*str, default: 'relu'*) – Name of the activation function used for the auto-encoder. Supported functions: ['sigmoid', 'tanh', 'elu', 'relu', 'relu6', 'leaky_relu', 'identity']

- **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.
- **corruption_rate** (*float, optional, default: 0.3*) – The corruption ratio for SDAE.
- **dropout_rate** (*float, optional, default: 0.1*) – The probability that each element is removed in dropout of SDAE.
- **batch_size** (*int, optional, default: 128*) – The batch size for SGD.
- **name** (*string, optional, default: 'CDR'*) – 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}

  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.
- **seed** (*int, optional, default: None*) – Random seed for weight initialization.

References

Collaborative Deep Ranking: A Hybrid Pair-Wise Recommendation Algorithm with Implicit Feedback Ying H., Chen L., Xiong Y., Wu J. (2016)

fit(*train_set*, *val_set=None*)

Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

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

Parameters

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

2.15 Collaborative Ordinal Embedding (COE)

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*, *val_set=None*)
Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

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

Parameters

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

2.16 Convolutional Matrix Factorization (ConvMF)

class cornac.models.conv_mf.recom_convmf.**ConvMF**(*give_item_weight=True*, *cnn_epochs=5*, *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*, *seed=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*, *val_set=None*)
Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

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

Parameters

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

2.17 Spherical k-means (Skmeans)

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*, *val_set=None*)
Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

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

Parameters

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

2.18 Visual Bayesian Personalized Ranking (VBPR)

class cornac.models.vbpr.recom_vbpr.**VBPR**(*name='VBPR'*, *k=10*, *k2=10*, *n_epochs=50*, *batch_size=100*, *learning_rate=0.005*, *lambda_w=0.01*, *lambda_b=0.01*, *lambda_e=0.0*, *use_gpu=False*, *trainable=True*, *verbose=True*, *init_params=None*, *seed=None*)

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).
- **verbose** (*boolean, optional, default: True*) – When True, running logs are displayed.
- **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}
- **seed** (*int, optional, default: None*) – Random seed for weight initialization.

References

- He, R., & McAuley, J. (2016). VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback.

fit(*train_set*, *val_set=None*)

Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.

- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

score(*user_idx*, *item_idx=None*)

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

Parameters

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

2.19 Collaborative Deep Learning (CDL)

class cornac.models.cdl.recom_cdl.**CDL**(*name='CDL'*, *k=50*, *autoencoder_structure=None*, *act_fn='relu'*, *lambda_u=0.1*, *lambda_v=100*, *lambda_w=0.1*, *lambda_n=1000*, *a=1*, *b=0.01*, *corruption_rate=0.3*, *learning_rate=0.001*, *vocab_size=8000*, *dropout_rate=0.1*, *batch_size=128*, *max_iter=100*, *trainable=True*, *verbose=True*, *init_params=None*, *seed=None*)

Collaborative Deep Learning.

Parameters

- **name** (*string, default: 'CDL'*) – The name of the recommender model.
- **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.
- **autoencoder_structure** (*list, default: None*) – The number of neurons of encoder/decoder layer for SDAE. For example, autoencoder_structure = [200], the SDAE structure will be [vocab_size, 200, k, 200, vocab_size]
- **act_fn** (*str, default: 'relu'*) – Name of the activation function used for the auto-encoder. Supported functions: [‘sigmoid’, ‘tanh’, ‘elu’, ‘relu’, ‘relu6’, ‘leaky_relu’, ‘identity’]
- **learning_rate** (*float, optional, default: 0.001*) – The learning rate for AdamOptimizer.
- **vocab_size** (*int, default: 8000*) – The size of text input for the SDAE.
- **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.
- **corruption_rate** (*float, optional, default: 0.3*) – The corruption ratio for input text of the SDAE.
- **dropout_rate** (*float, optional, default: 0.1*) – The probability that each element is removed in dropout of SDAE.
- **batch_size** (*int, optional, default: 100*) – The batch size for SGD.
- **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}

  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.
- **seed** (*int, optional, default: None*) – Random seed for weight initialization.

References

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

fit(*train_set*, *val_set=None*)
Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

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

Parameters

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

- **item_idx** (*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

2.20 Hierarchical Poisson Factorization (HPF)

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, val_set=None*)

Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.

- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

score(*user_idx*, *item_idx=None*)

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

Parameters

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

2.21 Explicit Factor Model (EFM)

class cornac.models.efm.recom_efm.**EFM**

Explict Factor Models

Parameters

- **num_explicit_factors** (*int, optional, default: 40*) – The dimension of the explicit factors.
- **num_latent_factors** (*int, optional, default: 60*) – The dimension of the latent factors.
- **num_most_cared_aspects** (*int, optional, default: 15*) – The number of most cared aspects for each user.
- **rating_scale** (*float, optional, default: 5.0*) – The maximum rating score of the dataset.
- **alpha** (*float, optional, default: 0.85*) – Trace off factor for constructing ranking score.
- **lambda_x** (*float, optional, default: 1*) – The regularization parameter for user aspect attentions.
- **lambda_y** (*float, optional, default: 1*) – The regularization parameter for item aspect qualities.
- **lambda_u** (*float, optional, default: 0.01*) – The regularization parameter for user and item explicit factors.
- **lambda_h** (*float, optional, default: 0.01*) – The regularization parameter for user and item latent factors.
- **lambda_v** (*float, optional, default: 0.01*) – The regularization parameter for V.
- **use_item_aspect_popularity** (*boolean, optional, default: True*) – When False, item aspect frequency is omitted from item aspect quality computation formular. Specifically, $Y_{ij} = 1 + \frac{N-1}{1+e^{-s_{ij}}}$ if p_i is reviewed on feature F_j

- **max_iter** (*int, optional, default: 100*) – Maximum number of iterations or the number of epochs.
- **name** (*string, optional, default: 'EFM'*) – The name of the recommender model.
- **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 is not trained and Cornac assumes that the model already pre-trained (U1, U2, V, H1, and H2 are not None).
- **verbose** (*boolean, optional, default: False*) – When True, running logs are displayed.
- **init_params** (*dictionary, optional, default: {}*) – List of initial parameters, e.g., init_params = {'U1':U1, 'U2':U2, 'V':V, 'H1':H1, 'H2':H2}

  U1: ndarray, shape (n_users, n_explicit_factors) The user explicit factors, optional initialization via init_params.

  U2: ndarray, shape (n_ratings, n_explicit_factors) The item explicit factors, optional initialization via init_params.

  V: ndarray, shape (n_aspects, n_explict_factors) The aspect factors, optional initialization via init_params.

  H1: ndarray, shape (n_users, n_latent_factors) The user latent factors, optional initialization via init_params.

  H2: ndarray, shape (n_ratings, n_latent_factors) The item latent factors, optional initialization via init_params.
- **seed** (*int, optional, default: None*) – Random seed for weight initialization.

References

Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval (SIGIR '14). ACM, New York, NY, USA, 83-92. DOI: https://doi.org/10.1145/2600428.2609579

fit

Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

rank

Rank all test items for a given user.

Parameters

- **user_idx** (*int, required*) – The index of the user for whom to perform item raking.
- **item_indices** (*1d array, optional, default: None*) – A list of candidate item indices to be ranked by the user. If *None*, list of ranked known item indices and their scores will be returned

Returns

- Tuple of *item_rank*, and *item_scores*. The order of values
- *in item_scores are corresponding to the order of their ids in item_ids*

score

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

Parameters

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

2.22 Social Bayesian Personalized Ranking (SBPR)

class cornac.models.sbpr.recom_sbpr.**SBPR**

Social 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 num_threads=0, all CPU cores will be utilized. If seed is not None, num_threads=1 to remove randomness from parallelization.
- **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}
- **seed** (*int, optional, default: None*) – Random seed for weight initialization. If specified, training will take longer because of single-thread (no parallelization).

References

- Zhao, T., McAuley, J., & King, I. (2014, November). Leveraging social connections to improve personalized ranking for collaborative filtering. CIKM 2014 (pp. 261-270).

fit

Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

score

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

Parameters

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

2.23 Hidden Factors and Hidden Topics (HFT)

class cornac.models.hft.recom_hft.**HFT**(*name='HFT', k=10, max_iter=50, grad_iter=50, lambda_text=0.1, l2_reg=0.001, vocab_size=8000, init_params=None, trainable=True, verbose=True, seed=None*)

Hidden Factors and Hidden Topics

Parameters

- **name** (*string, default: 'HFT'*) – The name of the recommender model.
- **k** (*int, optional, default: 10*) – The dimension of the latent factors.
- **max_iter** (*int, optional, default: 50*) – Maximum number of iterations for EM.
- **grad_iter** (*int, optional, default: 50*) – Maximum number of iterations for L-BFGS.
- **lambda_text** (*float, default: 0.1*) – Weight of corpus likelihood in objective function.
- **l2_reg** (*float, default: 0.001*) – Regularization for user item latent factors.
- **vocab_size** (*int, optional, default: 8000*) – Size of vocabulary for review text.

- **init_params** (*dictionary, optional, default: None*) – List of initial parameters, e.g., init_params = {'alpha': alpha, 'beta_u': beta_u, 'beta_i': beta_i, 'gamma_u': gamma_u, 'gamma_v': gamma_v}

  alpha: float Model offset, optional initialization via init_params.

  beta_u: ndarray. shape (n_user, 1) User biases, optional initialization via init_params.

  beta_u: ndarray. shape (n_item, 1) Item biases, optional initialization via init_params.

  gamma_u: ndarray, shape (n_users,k) The user latent factors, optional initialization via init_params.

  gamma_v: ndarray, shape (n_items,k) The item latent factors, optional initialization via init_params.

- **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.
- **seed** (*int, optional, default: None*) – Random seed for weight initialization.

References

Julian McAuley, Jure Leskovec. "Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text" RecSys '13 Proceedings of the 7th ACM conference on Recommender systems Pages 165-172

fit (*train_set*, *val_set=None*)
Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

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

Parameters

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

2.24 Collaborative Topic Modeling (CTR)

class cornac.models.ctr.recom_ctr.**CTR**(*name='CTR'*, *k=200*, *lambda_u=0.01*, *lambda_v=0.01*, *eta=0.01*, *a=1*, *b=0.01*, *max_iter=100*, *trainable=True*, *verbose=True*, *init_params=None*, *seed=None*)

Collaborative Topic Regression.

Parameters

- **name** (*string, default: 'CTR'*) – The name of the recommender model.
- **k** (*int, optional, default: 200*) – The dimension of the latent factors.
- **max_iter** (*int, optional, default: 100*) – Maximum number of iterations or the number of epochs for SGD.
- **lambda_u** (*float, optional, default: 0.01*) – The regularization parameter for users.
- **lambda_v** (*float, optional, default: 0.01*) – The regularization parameter for items.
- **a** (*float, optional, default: 1*) – The confidence of observed ratings.
- **b** (*float, optional, default: 0.01*) – The confidence of unseen ratings.
- **eta** (*float, optional, default: 0.01*) – Added value for smoothing phi.
- **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}

  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.
- **seed** (*int, optional, default: None*) – Random seed for weight initialization.

References

Wang, Chong, and David M. Blei. "Collaborative topic modeling for recommending scientific articles." Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2011.

fit(*train_set*, *val_set=None*)

Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

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

Parameters

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

2.25 Baseline Only

cornac.models.baseline_only.**recom_bo**
alias of *cornac.models.baseline_only.recom_bo*

2.26 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 num_threads=0, all CPU cores will be utilized. If seed is not None, num_threads=1 to remove randomness from parallelization.
- **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}
- **seed** (*int, optional, default: None*) – Random seed for weight initialization. If specified, training will take longer because of single-thread (no parallelization).

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** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

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

Parameters

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

2.27 Global Average (GlobalAvg)

class cornac.models.global_avg.recom_global_avg.**GlobalAvg**(*name='GlobalAvg'*)
Global Average baseline for rating prediction. Rating predictions equal to average rating of training data (not personalized).

Parameters **name** (*string, default: 'GlobalAvg'*) – The name of the recommender model.

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

Parameters

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

2.28 Matrix Factorization (MF)

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.
- **num_threads** (*int, optional, default: 0*) – Number of parallel threads for training. If num_threads=0, all CPU cores will be utilized. If seed is not None, num_threads=1 to remove randomness from parallelization.
- **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}
- **seed** (*int, optional, default: None*) – Random seed for weight initialization. If specified, training will take longer because of single-thread (no parallelization).

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** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

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

Parameters

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

2.29 Most Popular (MostPop)

class cornac.models.most_pop.recom_most_pop.**MostPop**(*name='MostPop'*)

Most Popular. Item are recommended based on their popularity (not personalized).

Parameters **name** (*string, default: 'MostPop'*) – The name of the recommender model.

fit(*train_set*, *val_set=None*)

Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns **self**

Return type object

score(*user_idx*, *item_idx=None*)

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

Parameters

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

2.30 Non-negative Matrix Factorization (NMF)

class cornac.models.nmf.recom_nmf.**NMF**

Non-negative Matrix Factorization

Parameters

- **k** (*int, optional, default: 15*) – The dimension of the latent factors.
- **max_iter** (*int, optional, default: 50*) – Maximum number of iterations or the number of epochs for SGD.

- **learning_rate** (*float, optional, default: 0.005*) – The learning rate.
- **lambda_u** (*float, optional, default: 0.06*) – The regularization parameter for user factors U.
- **lambda_v** (*float, optional, default: 0.06*) – The regularization parameter for item factors V.
- **lambda_bu** (*float, optional, default: 0.02*) – The regularization parameter for user biases Bu.
- **lambda_bi** (*float, optional, default: 0.02*) – The regularization parameter for item biases Bi.
- **use_bias** (*boolean, optional, default: False*) – When True, user, item, and global biases are used.
- **num_threads** (*int, optional, default: 0*) – Number of parallel threads for training. If num_threads=0, all CPU cores will be utilized. If seed is not None, num_threads=1 to remove randomness from parallelization.
- **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, 'mu': global_mean}
- **seed** (*int, optional, default: None*) – Random seed for weight initialization. If specified, training will take longer because of single-thread (no parallelization).

References

- Lee, D. D., & Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In Advances in neural information processing systems (pp. 556-562).
- Takahashi, N., Katayama, J., & Takeuchi, J. I. (2014). A generalized sufficient condition for global convergence of modified multiplicative updates for NMF. In Proceedings of 2014 International Symposium on Nonlinear Theory and its Applications (pp. 44-47).

fit

Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

score

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

Parameters

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

2.31 Probabilitic Matrix Factorization (PMF)

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={}, seed=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: {}*) – 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.
- **seed** (*int, optional, default: None*) – Random seed for parameters initialization.

References

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

fit (*train_set*, *val_set=None*)
Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

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

Parameters

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

2.32 Singular Value Decomposition (SVD)

class cornac.models.svd.recom_svd.**SVD**(*name='SVD'*, *k=10*, *max_iter=20*, *learning_rate=0.01*, *lambda_reg=0.02*, *early_stop=False*, *num_threads=0*, *trainable=True*, *verbose=False*, *init_params=None*, *seed=None*)
Singular Value Decomposition (SVD). The implementation is based on Matrix Factorization with biases.

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.
- **early_stop** (*boolean, optional, default: False*) – When True, delta loss will be checked after each iteration to stop learning earlier.
- **num_threads** (*int, optional, default: 0*) – Number of parallel threads for training. If num_threads=0, all CPU cores will be utilized. If seed is not None, num_threads=1 to remove randomness from parallelization.
- **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}
- **seed** (*int, optional, default: None*) – Random seed for weight initialization. If specified, training will take longer because of single-thread (no parallelization).

References

- Koren, Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In SIGKDD, pp. 426-434. 2008.
- Koren, Y. Factor in the neighbors: Scalable and accurate collaborative filtering. In TKDD, 2010.

2.33 Social Recommendation using PMF (SoRec)

class cornac.models.sorec.recom_sorec.**SoRec**(*name='SoRec', k=5, max_iter=100, learning_rate=0.001, lamda_c=10, lamda=0.001, gamma=0.9, weight_link=True, trainable=True, verbose=False, init_params={}, seed=None*)

Social recommendation using 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.
- **lamda_c** (*float, optional, default: 10*) – The parameter balancing the information from the user-item rating matrix and the user social network.
- **weight_link** (*boolean, optional, default: True*) – When true the social network links are weighted according to eq. (4) in the original paper.
- **name** (*string, optional, default: 'SOREC'*) – 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, V and Z 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, 'Z':Z}.

  U: a ndarray of shape (n_users, k) Containing the user latent factors.

  V: a ndarray of shape (n_items, k) Containing the item latent factors.

Z: a ndarray of shape (n_users, k) Containing the social network latent factors.

- **seed** (*int, optional, default: None*) – Random seed for parameters initialization.

References

- H. Ma, H. Yang, M. R. Lyu, and I. King. SoRec:Social recommendation using probabilistic matrix factorization. CIKM '08, pages 931–940, Napa Valley, USA, 2008.

fit(*train_set*, *val_set=None*)

Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

score(*user_idx*, *item_idx=None*)

Predict the scores/ratings of a user for an item. :param user_idx: The index of the user for whom to perform score prediction. :type user_idx: int, required :param item_idx: 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

2.34 Weighted Matrix Factorization (WMF)

class cornac.models.wmf.recom_wmf.**WMF**(*name='WMF'*, *k=200*, *lambda_u=0.01*, *lambda_v=0.01*, *a=1*, *b=0.01*, *learning_rate=0.001*, *batch_size=128*, *max_iter=100*, *trainable=True*, *verbose=True*, *init_params=None*, *seed=None*)

Weighted Matrix Factorization.

Parameters

- **name** (*string, default: 'WMF'*) – The name of the recommender model.
- **k** (*int, optional, default: 200*) – 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 AdamOptimizer.
- **lambda_u** (*float, optional, default: 0.01*) – The regularization parameter for users.

- **lambda_v** (*float, optional, default: 0.01*) – The regularization parameter for items.
- **a** (*float, optional, default: 1*) – The confidence of observed ratings.
- **b** (*float, optional, default: 0.01*) – The confidence of unseen ratings.
- **batch_size** (*int, optional, default: 128*) – The batch size for SGD.
- **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}

  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.
- **seed** (*int, optional, default: None*) – Random seed for weight initialization.

References

- Hu, Y., Koren, Y., & Volinsky, C. (2008, December). Collaborative filtering for implicit feedback datasets. In 2008 Eighth IEEE International Conference on Data Mining (pp. 263-272).
- Pan, R., Zhou, Y., Cao, B., Liu, N. N., Lukose, R., Scholz, M., & Yang, Q. (2008, December). One-class collaborative filtering. In 2008 Eighth IEEE International Conference on Data Mining (pp. 502-511).

fit (*train_set*, *val_set=None*)
Fit the model to observations.

Parameters

- **train_set** (*cornac.data.Dataset*, required) – User-Item preference data as well as additional modalities.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – User-Item preference data for model selection purposes (e.g., early stopping).

Returns self

Return type object

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

Parameters

- **user_idx** (*int, required*) – The index of the user for whom to perform score prediction.
- **item_idx** (*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 3

Metrics

3.1 Rating

3.1.1 Mean Absolute Error (MAE)

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

name
Name of the measure.

Type string, value: 'MAE'

3.1.2 Mean Squared Error (MSE)

class cornac.metrics.**MSE**
Mean Squared Error.

name
Name of the measure.

Type string, value: 'MSE'

3.1.3 Root Mean Squared Error (RMSE)

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

name
Name of the measure.

Type string, value: 'RMSE'

3.2 Ranking

3.2.1 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

3.2.2 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.

3.2.3 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

3.2.4 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.

3.2.5 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

3.2.6 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.

3.2.7 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.

CHAPTER 4

Evaluation methods

4.1 Base Method

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

Base Evaluation Method

Parameters

- **data** (*array-like, required*) – Raw preference data in the triplet format [(user_id, item_id, rating_value)].
- **rating_threshold** (*float, optional, default: 1.0*) – Threshold used to binarize rating values into positive or negative feedback for model evaluation using ranking metrics (rating metrics are not affected).
- **seed** (*int, optional, default: None*) – Random seed for reproducibility.
- **exclude_unknowns** (*bool, optional, default: True*) – If *True*, unknown users and items will be ignored during model 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, optional, default: False*) – Evaluation mode. Whether results are averaging based on number of users or number of ratings.

Returns **res**

Return type cornac.experiment.Result

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

Constructing evaluation method given data.

Parameters

- **train_data** (*array-like*) – Training data
- **test_data** (*array-like*) – Test data
- **val_data** (*array-like, optional, default: None*) – Validation data
- **fmt** (*str, default: 'UIR'*) – Format of the input data. Currently, we are supporting:

  'UIR': User, Item, Rating 'UIRT': User, Item, Rating, Timestamp
- **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.
- **seed** (*int, optional, default: None*) – Random seed for reproduce the splitting.
- **verbose** (*bool, default: False*) – The verbosity flag.

Returns **method** – Evaluation method object.

Return type <cornac.eval_methods.BaseMethod>

cornac.eval_methods.base_method.**ranking_eval**(*model*, *metrics*, *train_set*, *test_set*, *val_set=None*, *rating_threshold=1.0*, *exclude_unknowns=True*, *verbose=False*)

Evaluate model on provided ranking metrics.

Parameters

- **model** (cornac.models.Recommender, required) – Recommender model to be evaluated.
- **metrics** (iterable, required) – List of rating metrics cornac.metrics.RankingMetric.
- **train_set** (*cornac.data.Dataset*, required) – Dataset to be used for model training. This will be used to exclude observations already appeared during training.
- **test_set** (*cornac.data.Dataset*, required) – Dataset to be used for evaluation.
- **val_set** (*cornac.data.Dataset*, optional, default: None) – Dataset to be used for model selection. This will be used to exclude observations already appeared during validation.
- **rating_threshold** (*float, optional, default: 1.0*) – The threshold to convert ratings into positive or negative feedback.
- **exclude_unknowns** (*bool, optional, default: True*) – Ignore unknown users and items during evaluation.
- **verbose** (*bool, optional, default: False*) – Output evaluation progress.

Returns

res –

Tuple of two lists:

- average result for each of the metrics
- average result per user for each of the metrics

Return type (List, List)

cornac.eval_methods.base_method.**rating_eval**(*model*, *metrics*, *test_set*, *user_based=False*)

Evaluate model on provided rating metrics.

Parameters

- **model** (cornac.models.Recommender, required) – Recommender model to be evaluated.
- **metrics** (iterable, required) – List of rating metrics cornac.metrics.RatingMetric.
- **test_set** (*cornac.data.Dataset*, required) – Dataset to be used for evaluation.
- **user_based** (*bool, optional, default: False*) – Evaluation mode. Whether results are averaging based on number of users or number of ratings.

Returns

res –

Tuple of two lists:

- average result for each of the metrics
- average result per user for each of the metrics

Return type (List, List)

4.2 Ratio Split

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

Splitting data into training, validation, and test sets based on provided sizes. Data is always shuffled before split.

Parameters

- **data** (*array-like, required*) – Raw preference data in the triplet format [(user_id, item_id, rating_value)].
- **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: 1.0*) – Threshold used to binarize rating values into positive or negative feedback for model evaluation using ranking metrics (rating metrics are not affected).
- **seed** (*int, optional, default: None*) – Random seed for reproducibility.

- **exclude_unknowns** (*bool, optional, default: True*) – If *True*, unknown users and items will be ignored during model evaluation.
- **verbose** (*bool, optional, default: False*) – Output running log.

4.3 Cross Validation

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

Cross Validation Evaluation Method.

Parameters

- **data** (*array-like, required*) – Raw preference data in the triplet format [(user_id, item_id, rating_value)].
- **n_folds** (*int, optional, default: 5*) – The number of folds for cross validation.
- **rating_threshold** (*float, optional, default: 1.0*) – Threshold used to binarize rating values into positive or negative feedback for model evaluation using ranking metrics (rating metrics are not affected).
- **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.
- **seed** (*int, optional, default: None*) – Random seed for reproducibility.
- **exclude_unknowns** (*bool, optional, default: True*) – If *True*, unknown users and items will be ignored during model 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, optional, default: False*) – Evaluation mode. Whether results are averaging based on number of users or number of ratings.

Returns res

Return type cornac.experiment.Result

CHAPTER 5

Experiment

5.1 Experiment

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

Experiment Class

Parameters

- **eval_method** (<cornac.eval_methods.BaseMethod>, required) – The evaluation method (e.g., RatioSplit).
- **models** (array of <cornac.models.Recommender>, required) – A collection of recommender models to evaluate, e.g., [C2PF, HPF, PMF].
- **metrics** (array of :obj:{<*cornac.metrics.RatingMetric*>, <*cornac.metrics.RankingMetric*>}, required) – A collection of metrics to use to evaluate the recommender models, e.g., [NDCG, MRR, Recall].
- **user_based** (*bool, optional, default: True*) – This parameter is only useful if you are considering rating metrics. When True, first the average performance for every user is computed, then the obtained values are averaged to return the final result. If *False*, results will be averaged over the number of ratings.
- **result** (array of <cornac.experiment.result.Result>, default: None) – This attribute contains the results per-model of your experiment, initially it is set to None.

5.2 Result

class cornac.experiment.result.**Result**(*model_name*, *metric_avg_results*, *metric_user_results*)

Result Class for a single model

Parameters

- **model_name** (*string, required*) – The name of the recommender model.
- **metric_avg_results** (OrderedDict, required) – A dictionary containing the average result per-metric.
- **metric_user_results** (defaultdict, required) – A dictionary containing the average result per-user across different metrics.

CHAPTER 6

Built-in datasets

6.1 Amazon Clothing

This data is built based on the Amazon datasets provided by Julian McAuley @ http://jmcauley.ucsd.edu/data/amazon/. We make sure all items having three types of auxiliary data: text, image, and context (items appearing together).

`cornac.datasets.amazon_clothing.load_context(reader: cornac.data.reader.Reader = None) → List`

Load the item-item interactions

Parameters **reader** (*obj:cornac.data.Reader*, default: None) – Reader object used to read the data.

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

Return type array-like

`cornac.datasets.amazon_clothing.load_image()`

Load the item image in the form of visual features (extracted from pre-trained CNN)

Returns

- **features** (*numpy.ndarray*) – Feature matrix with shape (n, 4096) with n is the number of items.
- **item_ids** (*List*) – List of item ids aligned with indices in *features*.

`cornac.datasets.amazon_clothing.load_rating(reader: cornac.data.reader.Reader = None) → List`

Load the user-item ratings

Parameters **reader** (*obj:cornac.data.Reader*, default: None) – Reader object used to read the data.

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

Return type array-like

cornac.datasets.amazon_clothing.**load_text**()
Load the item text descriptions

Returns

- **texts** (*List*) – List of text documents, one per item.
- **ids** (*List*) – List of item ids aligned with indices in *texts*.

6.2 Amazon Office

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**(*reader: cornac.data.reader.Reader = None*) → List
Load the item-item interactions

Parameters **reader** (*obj:cornac.data.Reader*, default: None) – Reader object used to read the data.

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

Return type array-like

cornac.datasets.amazon_office.**load_rating**(*reader: cornac.data.reader.Reader = None*) → List
Load the user-item ratings

Parameters **reader** (*obj:cornac.data.Reader*, default: None) – Reader object used to read the data.

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

Return type array-like

6.3 Amazon Toys and Games

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

cornac.datasets.amazon_toy.**load_rating**(*reader: cornac.data.reader.Reader = None*) → List
Load the user-item ratings

Parameters **reader** (*obj:cornac.data.Reader*, default: None) – Reader object used to read the data.

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

Return type array-like

cornac.datasets.amazon_toy.**load_sentiment**(*reader: cornac.data.reader.Reader = None*) → List
Load the user-item-sentiments The dataset was constructed by the method described in the reference paper.

Parameters **reader** (*obj:cornac.data.Reader*, default: None) – Reader object used to read the data.

Returns **data** – Data in the form of a list of tuples (user, item, [(aspect, opinion, sentiment), (aspect, opinion, sentiment), ...]).

Return type array-like

References

Gao, J., Wang, X., Wang, Y., & Xie, X. (2019). Explainable Recommendation Through Attentive Multi-View Learning. AAAI.

6.4 CiteULike

This dataset is mostly from the paper 'Collaborative topic modeling for recommending scientific articles' [Wang and Blei - KDD 2011]. It was further collected, named *citeulike-a*, and used in the paper 'Collaborative Topic Regression with Social Regularization' [Wang, Chen and Li - IJCAI 2013].

Link to the data: http://www.wanghao.in/CDL.htm

`cornac.datasets.citeulike.`**`load_data`**(*reader: cornac.data.reader.Reader = None*) → List
Load the implicit feedback between users and items

Parameters **`reader`** (*obj:cornac.data.Reader*, default: None) – Reader object used to read the data.

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

Return type array-like

`cornac.datasets.citeulike.`**`load_text`**()
Load item texts including tile and abstract joined together into one document per item.

Returns

- **texts** (*List*) – List of text documents, one per item.
- **ids** (*List*) – List of item ids aligned with indices in *texts*.

6.5 Epinions

Link to the dataset: http://www.trustlet.org/downloaded_epinions.html

`cornac.datasets.epinions.`**`load_data`**(*reader: cornac.data.reader.Reader = None*) → List
Load the rating feedback

Parameters **`reader`** (*obj:cornac.data.Reader*, default: None) – Reader object used to read the data.

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

Return type array-like

`cornac.datasets.epinions.`**`load_trust`**(*reader: cornac.data.reader.Reader = None*) → List
Load the trust data

Parameters **`reader`** (*obj:cornac.data.Reader*, default: None) – Reader object used to read the data.

Returns **data** – Data in the form of a list of tuples (source_user, target_item, trust_value).

Return type array-like

6.6 FilmTrust

Source: https://www.librec.net/datasets.html

cornac.datasets.filmtrust.**load_feedback**(*reader: cornac.data.reader.Reader = None*) → List

Load the user-item ratings, scale: [0.5,4]

Parameters **reader** (*obj:cornac.data.Reader*, default: None) – Reader object used to read the data.

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

Return type array-like

cornac.datasets.filmtrust.**load_trust**(*reader: cornac.data.reader.Reader = None*) → List

Load the user-user trust information (undirected network)

Parameters **reader** (*obj:cornac.data.Reader*, default: None) – Reader object used to read the data.

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

Return type array-like

6.7 MovieLens

Link to the data: https://grouplens.org/datasets/movielens/

cornac.datasets.movielens.**load_100k**(*fmt='UIR'*, *reader=None*)

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'*, *reader: cornac.data.reader.Reader = None*) → List

Load the MovieLens 1M dataset

Parameters

- **fmt** (*str, default: 'UIR'*) – Data format to be returned.
- **reader** (*obj:cornac.data.Reader*, default: None) – Reader object used to read the data.

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

- **texts** (*List*) – List of text documents, one per item.
- **ids** (*List*) – List of item ids aligned with indices in *texts*.

6.8 Netflix

Link to the data: https://www.kaggle.com/netflix-inc/netflix-prize-data/

cornac.datasets.netflix.**load_data**(*fmt='UIR'*, *reader: cornac.data.reader.Reader = None*) → List

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.
- **reader** (*obj:cornac.data.Reader*, default: None) – Reader object used to read the data.

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'*, *reader: cornac.data.reader.Reader = None*) → List

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.
- **reader** (*obj:cornac.data.Reader*, default: None) – Reader object used to read the data.

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

Return type array-like

6.9 Tradesy

Link to the 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**(*reader: cornac.data.reader.Reader = None*) → List

Load the feedback observations

Parameters **reader** (*obj:cornac.data.Reader*, default: None) – Reader object used to read the data.

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

- **features** (*numpy.ndarray*) – Feature matrix with shape (n, 4096) with n is the number of items.
- **item_ids** (*List*) – List of item ids aligned with indices in *features*.

Python Module Index

C

Index

D

E

F

G

H

I

S

T

U

V

W