Alpenglow Documentation

Release 0.1

MTA SZTAKI

Sep 19, 2018

Getting started:

CHAPTER 1

Introduction

Welcome to Alpenglow introduction!

Alpenglow is an open source recommender systems research framework, aimed at providing tools for rapid prototyping and evaluation of algorithms for streaming recommendation tasks.

The framework is composed of a large number of components written in C++ and a thin python API for combining them into reusable experiments, thus enabling ease of use and fast execution at the same time. The framework also provides a number of preconfigured experiments in the *alpenglow.experiments* package and various tools for evaluation, hyperparameter search, etc.

1.1 Requirements

Anaconda environment with Python >= 3.5

1.2 Installing

```
conda install -c conda-forge alpenglow
```

1.3 Installing from source on Linux

```
cd Alpenglow
conda install libgcc sip
conda install -c conda-forge eigen
pip install .
```

1.4 Development

- For faster recompilation, use `export CC="ccache cc"`
- To enable compilation on 4 threads for example, use `echo 4 > .parallel`
- Reinstall modified version using `pip install --upgrade --force-reinstall --no-deps .`
- To build and use in the current folder, `use pip install --upgrade --force-reinstall --no-deps -e .` and `export PYTHONPATH="$(pwd)/python:$PYTHONPATH"`

CHAPTER 2

Example usage

Sample dataset: http://info.ilab.sztaki.hu/~fbobee/alpenglow/alpenglow_sample_dataset

```
from alpenglow.experiments import FactorExperiment
from alpenglow.evaluation import DcgScore
import pandas as pd
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt

data = pd.read_csv("/path/to/sample_dataset")

factor_model_experiment = FactorExperiment(
    top_k=100,
    seed=254938879,
    dimension=10,
    learning_rate=0.14,
    negative_rate=100
)
fac_rankings = factor_model_experiment.run(data, verbose=True)
fac_rankings['dcg'] = DcgScore(fac_rankings)
fac_rankings['dcg'].groupby((fac_rankings['time']-fac_rankings['time'].min())//86400).
↪mean().plot()
plt.savefig("factor.png")
```

CHAPTER 3

Five minute tutorial

In this tutorial we are going to learn the basic concepts of using Alpenglow by evaluating various baseline models on real world data.

3.1 The data

You can find the dataset at http://info.ilab.sztaki.hu/~fbobee/alpenglow/alpenglow_sample_dataset. This is a processed version of the [30M dataset](http://info.ilab.sztaki.hu/~fbobee/alpenglow/recoded_online_id_artist_first_filtered), where we

- only keep users above a certain activity threshold
- only keep the first events of listening sessions
- recode the items so they represent artists instead of tracks

Let's start by importing standard packages and Alpenglow; and then reading the csv file using pandas. To avoid waiting too much for the experiments to complete, we limit the amount of records read to 200000.

```
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import alpenglow as ag

data = pd.read_csv('data', nrows=200000)
print(data.columns)
```

Output:

```
Index(['time', 'user', 'item', 'score', 'eval', 'category'], dtype='object')
```

To run online experiments, you will need time-series data of user-item interactions in similar format to the above. The only required columns are the `'user'` and `'item'` columns – the rest will be autofilled if missing. The most important columns are the following:

- **time**: integer, the timestamp of the record. Controls various things, like evaluation timeframes or batch learning epochs. Defaults to `range(0,len(data))` if missing.
- **user**: integer, the user the activity belongs to. This column is required.
- **item**: integer, the item the activity belongs to. This column is required.
- **score**: double, the score corresponding to the given record. This could be for example the rating of the item in the case of explicit recommendation. Defaults to constant `1`.
- **eval**: boolean, whether to run ranking-evaluation on the record. Defaults to constant `True`.

3.2 Our first model

Let's start by evaluating a very basic model on the dataset, the popularity model. To do this, we need to import the preconfigured experiment from the package `alpenglow.experimens`.

```
from alpenglow.experiments import PopularityExperiment
```

When creating an instance of the experiment, we can provide various configuration options and parameters.

```
pop_experiment = PopularityExperiment(
    top_k=100, # we are going to evaluate on top 100 ranking lists
    seed=12345, # for reproducibility, we provide a random seed
)
```

You can see the list available options of online experiments in the documentation of *alpenglow.OnlineExperiment* and the parameters of this particular experiment in the documentation of the specific implementation (in this case *alpenglow.experiments.PopularityExperiment*) or, failing that, in the source code of the given class.

Running the experiment on the data is as simple as calling `run(data)`. Multiple options can be provided at this point, for a full list, refer to the documentation of *alpenglow.OnlineExperiment.OnlineExperiment.run()*.

```
result = pop_experiment.run(data, verbose=True) #this might take a while
```

The `run()` method first builds the experiment out of C++ components according to the given parameters, then processes the data, training on it and evaluating the model at the same time. The returned object is a `pandas.DataFrame` object, which contains various information regarding the results of the experiment:

```
print(result.columns)
```

Output:

```
Index(['time', 'score', 'user', 'item', 'prediction', 'rank'], dtype='object')
```

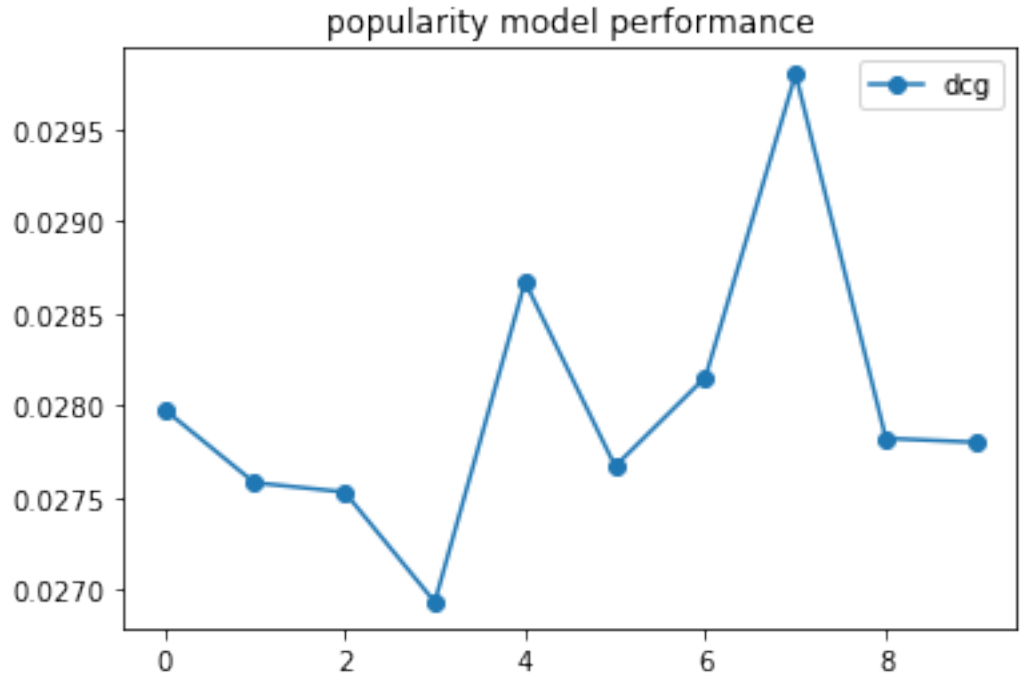
Prediction is the score estimate given by the model and rank is the rank of the item in the toplist generated by the model. If the item is not on the toplist, rank is `NaN`.

The easiest way interpret the results is by using a predefined evaluator, for example *alpenglow.evaluation.DcgScore*:

```
from alpenglow.evaluation import DcgScore
results['dcg'] = DcgScore(results)
```

The `DcgScore` class calculates the NDCG values for the given ranks and returns a `pandas.Series` object. This can be averaged and plotted easily to visualize the performance of the recommender model.

```
daily_avg_dcg = results['dcg'].groupby((results['time']-results['time'].min())//
↪86400).mean()
plt.plot(daily_avg_dcg,"o-", label="popularity")
plt.title('popularity model performance')
plt.legend()
```



Putting it all together:

```
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
from alpenglow.evaluation import DcgScore
from alpenglow.experiments import PopularityExperiment

data = pd.read_csv('data', nrows=200000)

pop_experiment = PopularityExperiment(
    top_k=100,
    seed=12345,
)
results = pop_experiment.run(data, verbose=True)
results['dcg'] = DcgScore(results)
daily_avg_dcg = results['dcg'].groupby((results['time']-results['time'].min())//
↪86400).mean()

plt.plot(daily_avg_dcg,"o-", label="popularity")
plt.title('popularity model performance')
plt.legend()
```

3.3 Matrix factorization, hyperparameter search

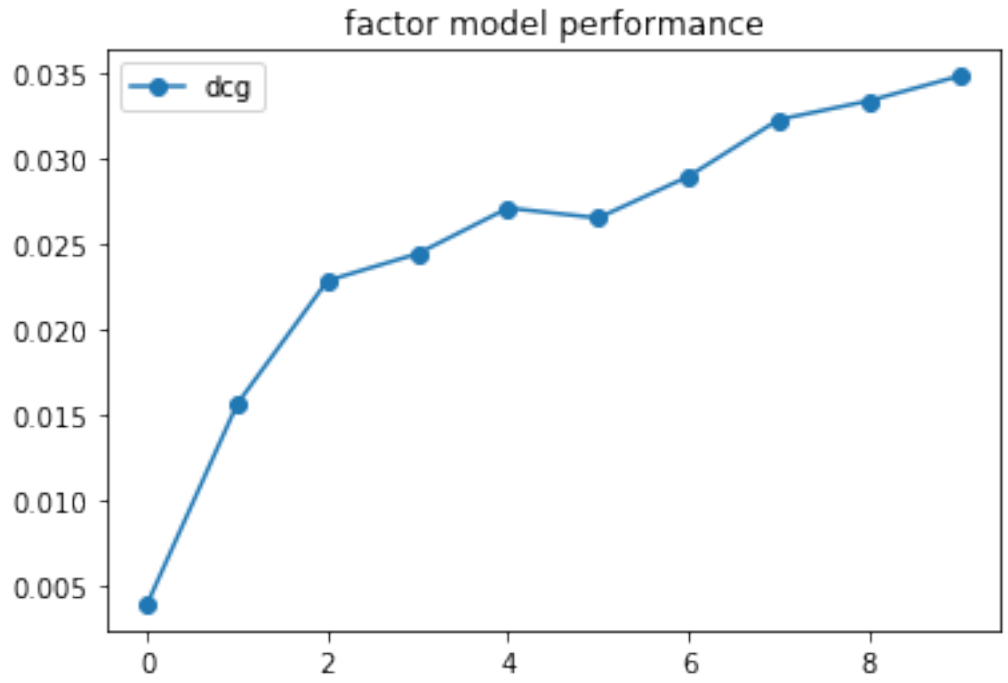
The *alpenglow.experiments.FactorExperiment* class implements a factor model, which is updated in an online fashion. After checking the documentation / source, we can see that the most relevant hyperparameters for this model are `dimension` (the number of latent factors), `learning_rate`, `negative_rate` and `regularization_rate`. For this experiment, we are leaving the factor dimension at the default value of 10, and we don't need regularization, so we'll leave it at its default (0) as well. We will find the best negative rate and learning rate using grid search.

We can run the `FactorModelExperiment` similarly to the popularity model:

```
from alpenglow.experiments import FactorExperiment

mf_experiment = FactorExperiment(
    top_k=100,
)
mf_results = mf_experiment.run(data, verbose=True)
mf_results['dcg'] = DcgScore(mf_results)
mf_daily_avg = mf_results['dcg'].groupby((mf_results['time']-mf_results['time'].
↪min())//86400).mean()

plt.plot(mf_daily_avg,"o-", label="factorization")
plt.title('factor model performance')
plt.legend()
```



The default parameters are chosen to perform generally well. However, the best choice always depends on the task at hand. To find the best values for this particular dataset, we can use Alpenglow's built in multithreaded hyperparameter search tool: `alpenglow.ThreadedParameterSearch`.

```
mf_parameter_search = ag.ThreadedParameterSearch(mf_experiment, DcgScore, threads=4)
mf_parameter_search.set_parameter_values('negative_rate', np.linspace(10, 100, 4))
```

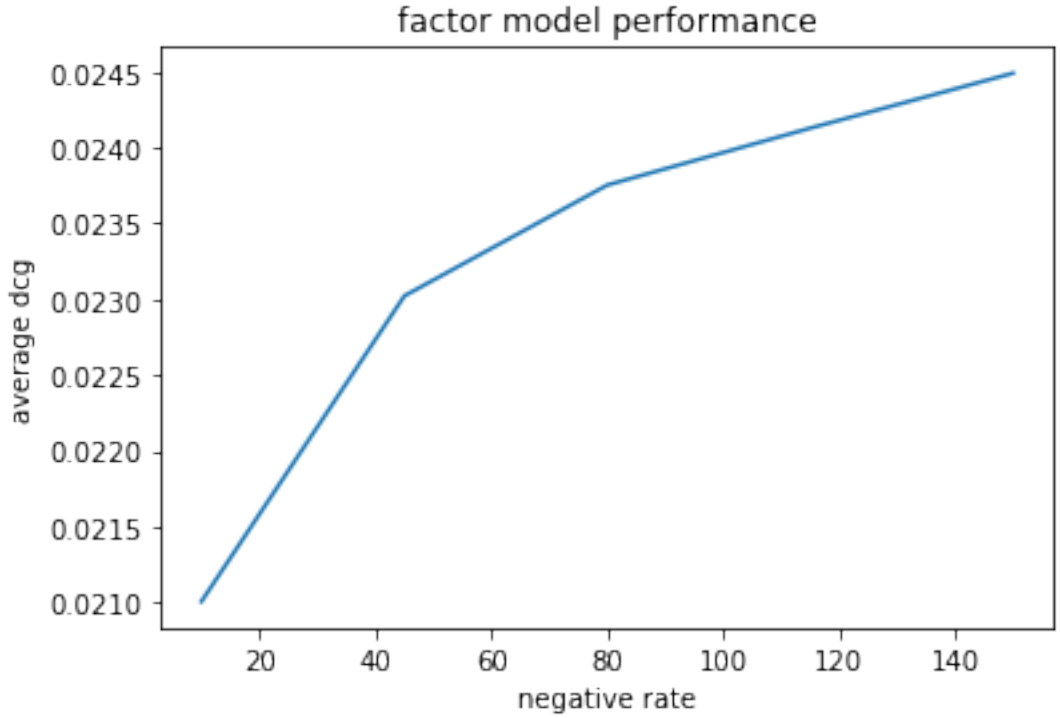
The `ThreadedParameterSearch` instance wraps around an `OnlineExperiment` instance. With each call to

the function `set_parameter_values`, we can set a new dimension for the grid search, which runs the experiments in parallel accoring to the given `threads` parameter. We can start the hyperparameter search similar to the experiment itself: by calling `run()`.

```
neg_rate_scores = mf_parameter_search.run(data, verbose=False)
```

The result of the search is a pandas DataFrame, with columns representing the given parameters and the score itself.

```
plt.plot(neg_rate_scores['negative_rate'], neg_rate_scores['DcgScore'])
plt.ylabel('average dcg')
plt.xlabel('negative rate')
plt.title('factor model performance')
```



CHAPTER 4

Further reading

If you want to get familiar with Alpenglow quickly, we collected a list of resources for you to read.

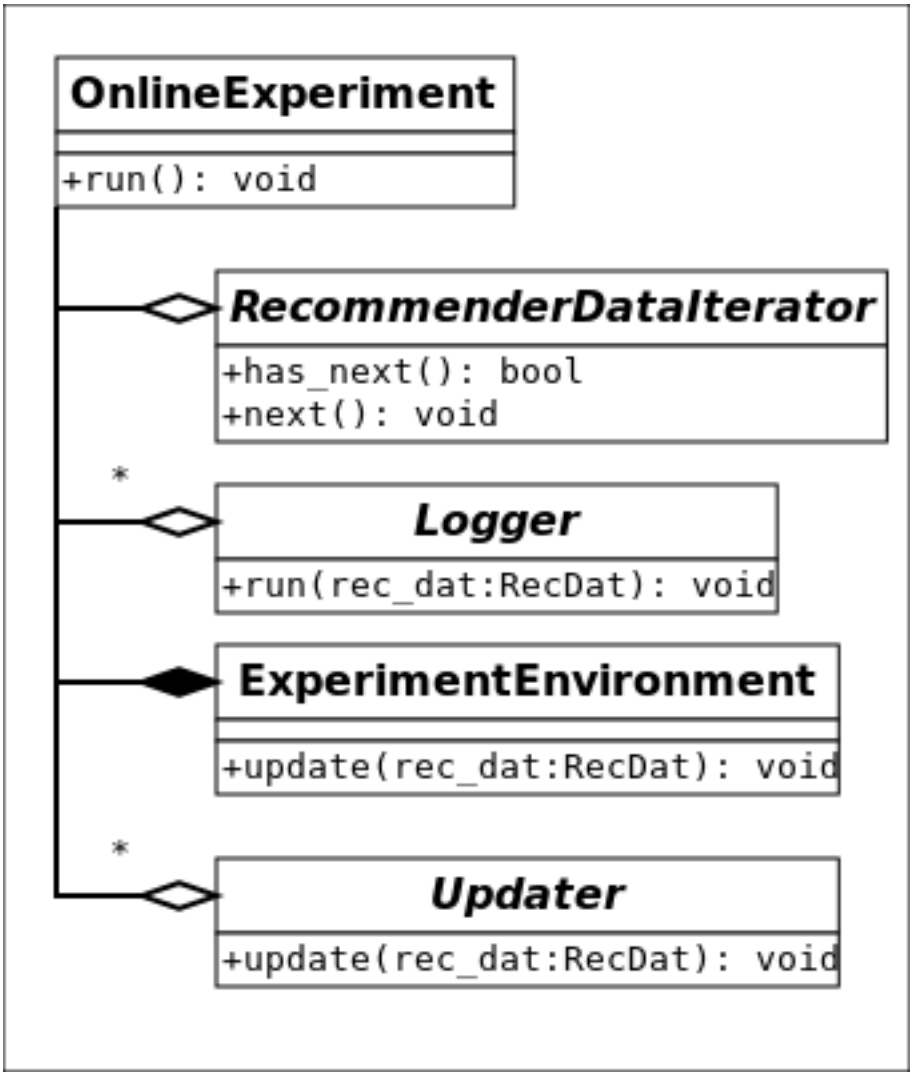
1. The documentation of *alpenglow.OnlineExperiment*. This describes basic information about running online experiments with alpenglow, and the parameters that are shared between all implementations.
2. The documentation of implemented experiments in the `alpenglow.experimens` package, which briefly describe the algorithms themselves and their parameters.
3. The documentation of *alpenglow.offline.OfflineModel*, which describes how to use Alpenglow for traditional, scikit-learn style machine learning.
4. The documentation of implemented offline models in the *alpenglow.offline.models* package.
5. Any pages from the the General section of this documentation

CHAPTER 5

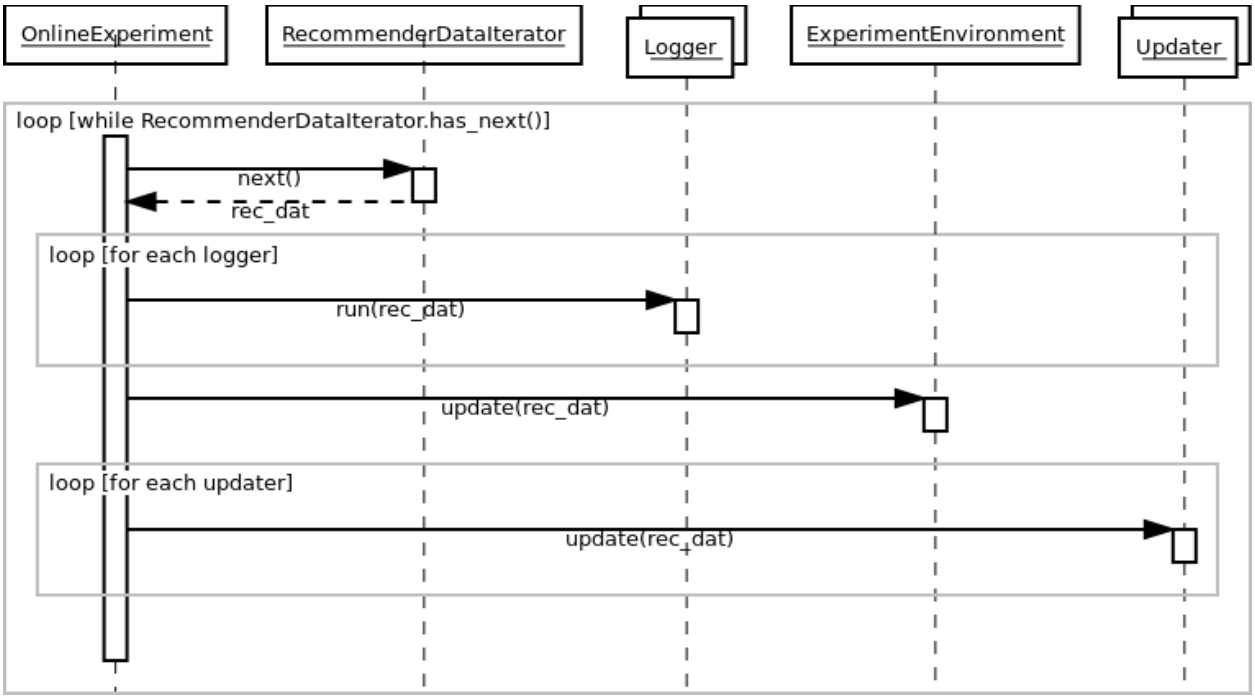
The anatomy of an Alpenglow experiment

The online experiment runs on a time series of events. The system performs two steps for each event. First, it evaluates the recommender, using the event as an evaluation sample. Second, using the event as training data, allows the recommender model to update itself.

In our C++ implementation, the central class is *alpenglow.cpp.OnlineExperiment* that manages the process described above. The data, the evaluators and the training algorithms are set into this class, and they have to implement the appropriate interfaces.



The data must implement the interface *alpenglow.cpp.RecommenderDataIterator*. This class behaves like an iterator, but provides random access availability to the time series also. In the preconfigured experiments, we normally use *alpenglow.cpp.ShuffleIterator* that randomizes the order of events having identical timestamp. Use *alpenglow.cpp.SimpleIterator* to avoid shuffling.



While processing an event, we first treat it as an evaluation sample. The system passes the sample to *alpenglow.cpp.Logger* objects that are set into the experiment. Loggers can evaluate the model or log out any statistic for example. Loggers are not allowed to update the state of the model, even if they have non-const access to the model, that is the situation in many cases because of caching implemented in some models.

After evaluation, the model is allowed to use the sample as a training sample. First we update some common containers and statistics of *alpenglow.cpp.ExperimentEnvironment*. Model updating algorithms are organised into a chain, or more precisely into a DAG. You can add any number of *alpenglow.cpp.Updater* objects into the experiment, and the system will pass the positive sample to each of them. Some *alpenglow.cpp.Updater* implementations can accept other *alpenglow.cpp.Updater* objects and passes them further the samples, possibly completed with extra information (e.g. gradient value) or mixed with generated samples (e.g. generated negative samples).

CHAPTER 6

alpenglow package

6.1 Subpackages

6.1.1 alpenglow.evaluation package

Submodules

alpenglow.evaluation.DcgScore module

alpenglow.evaluation.DcgScore.**Dcg**(*rank*)

alpenglow.evaluation.DcgScore.**DcgScore**(*rankings*)

alpenglow.evaluation.MseScore module

alpenglow.evaluation.MseScore.**MseScore**(*rankings*)

alpenglow.evaluation.PrecisionScore module

alpenglow.evaluation.PrecisionScore.**Precision**(*rank*)

alpenglow.evaluation.PrecisionScore.**PrecisionScore**(*rankings*)

alpenglow.evaluation.RecallScore module

alpenglow.evaluation.RecallScore.**Recall**(*rank*, *top_k*)

alpenglow.evaluation.RecallScore.**RecallScore**(*rankings*, *top_k=None*)

alpenglow.evaluation.RrScore module

alpenglow.evaluation.RrScore.**Rr**(*rank*)

alpenglow.evaluation.RrScore.**RrScore**(*rankings*)
Reciprocial rank, see https://en.wikipedia.org/wiki/Mean_reciprocal_rank .

Module contents

6.1.2 alpenglow.experiments package

Submodules

alpenglow.experiments.ALSFactorExperiment module

class alpenglow.experiments.ALSFactorExperiment.**ALSFactorExperiment**(*dimension=10*, *begin_min=-0.01*, *begin_max=0.01*, *number_of_iterations=15*, *regularization_lambda=1e-3*, *alpha=40*, *implicit=1*, *clear_before_fit=1*, *period_length=86400*)

Bases: *alpenglow.OnlineExperiment.OnlineExperiment*

This class implements an online version of the well-known matrix factorization recommendation model [Koren2009] and trains it via Alternating Least Squares in a periodic fashion. The model is able to train on explicit data using traditional ALS, and on implicit data using the iALS algorithm [Hu2008].

Parameters

- **dimension** (*int*) – The latent factor dimension of the factormodel.
- **begin_min** (*double*) – The factors are initialized randomly, sampling each element uniformly from the interval (begin_min, begin_max).
- **begin_max** (*double*) – See begin_min.
- **number_of_iterations** (*int*) – The number of ALS iterations to perform in each period.
- **regularization_lambda** (*double*) – The coefficient for the L2 regularization term. See [Hu2008]. This number is multiplied by the number of non-zero elements of the user-item rating matrix before being used, to achieve similar magnitude to the one used in traditional SGD.
- **alpha** (*int*) – The weight coefficient for positive samples in the error formula. See [Hu2008].
- **implicit** (*int*) – Valued 1 or 0, indicating whether to run iALS or ALS.
- **clear_before_fit** (*int*) – Whether to reset the model after each period.

- **period_length** (*int*) – The period length in seconds.
- **timeframe_length** (*int*) – The size of historic time interval to iterate over at every batch model retrain. Leave at the default 0 to retrain on everything.

alpenglow.experiments.ALSOnlineFactorExperiment module

class alpenglow.experiments.ALSOnlineFactorExperiment.**ALSOnlineFactorExperiment**(*dimension=10*, *begin_min=-0.01*, *begin_max=0.01*, *number_of_iteration*, *regularization_*, *3*, *alpha=40*, *implicit=1*, *clear_before_fi*, *period_length=86*

Bases: *alpenglow.OnlineExperiment.OnlineExperiment*

Combines ALSFactorExperiment and FactorExperiment by updating the model periodically with ALS and continously with SGD.

Parameters

- **dimension** (*int*) – The latent factor dimension of the factormodel.
- **begin_min** (*double*) – The factors are initialized randomly, sampling each element uniformly from the interval (begin_min, begin_max).
- **begin_max** (*double*) – See begin_min.
- **number_of_iterations** (*double*) – Number of times to optimize the user and the item factors for least squares.
- **regularization_lambda** (*double*) – The coefficient for the L2 regularization term. See [Hu2008]. This number is multiplied by the number of non-zero elements of the user-item rating matrix before being used, to achieve similar magnitude to the one used in traditional SGD.
- **alpha** (*int*) – The weight coefficient for positive samples in the error formula. See [Hu2008].
- **implicit** (*int*) – Valued 1 or 0, indicating whether to run iALS or ALS.
- **clear_before_fit** (*int*) – Whether to reset the model after each period.
- **period_length** (*int*) – The period length in seconds.
- **timeframe_length** (*int*) – The size of historic time interval to iterate over at every batch model retrain. Leave at the default 0 to retrain on everything.
- **online_learning_rate** (*double*) – The learning rate used in the online stochastic gradient descent updates.

- **online_regularization_rate** (*double*) – The coefficient for the L2 regularization term for online update.
- **online_negative_rate** (*int*) – The number of negative samples generated after online each update. Useful for implicit recommendation.

alpenglow.experiments.AsymmetricFactorExperiment module

class alpenglow.experiments.AsymmetricFactorExperiment.**AsymmetricFactorExperiment**(*dimension=*
begin_min=
0.01,
be-
gin_max=0.
learn-
ing_rate=0.
reg-
u-
lar-
iza-
tion_rate=0.
neg-
a-
tive_rate=20
cu-
mu-
la-
tive_item_up
norm_type=
gamma=0.8

Bases: *alpenglow.OnlineExperiment.OnlineExperiment*

Implements the recommendation model introduced in [Paterek2007].

Parameters

- **dimension** (*int*) – The latent factor dimension of the factormodel.
- **begin_min** (*double*) – The factors are initialized randomly, sampling each element uniformly from the interval (begin_min, begin_max).
- **begin_max** (*double*) – See begin_min.
- **learning_rate** (*double*) – The learning rate used in the stochastic gradient descent updates.
- **regularization_rate** (*double*) – The coefficient for the L2 regularization term.
- **negative_rate** (*int*) – The number of negative samples generated after each update. Useful for implicit recommendation.
- **norm_type** (*str*) – Type of time decay; either “constant”, “exponential” or “disabled”.
- **gamma** (*double*) – Coefficient of time decay in the case of **norm_type** == “exponential”.

alpenglow.experiments.BatchAndOnlineFactorExperiment module

class alpenglow.experiments.BatchAndOnlineFactorExperiment.**BatchAndOnlineFactorExperiment**(

Bases: *alpenglow.OnlineExperiment.OnlineExperiment*

Combines BatchFactorExperiment and FactorExperiment by updating the model both in batch and continously.

Parameters

- **dimension** (*int*) – The latent factor dimension of the factormodel.
- **begin_min** (*double*) – The factors are initialized randomly, sampling each element uniformly from the interval (begin_min, begin_max).
- **begin_max** (*double*) – See begin_min.
- **batch_learning_rate** (*double*) – The learning rate used in the batch stochastic gradient descent updates.
- **batch_regularization_rate** (*double*) – The coefficient for the L2 regularization term for batch updates.
- **batch_negative_rate** (*int*) – The number of negative samples generated after each batch update. Useful for implicit recommendation.
- **timeframe_length** (*int*) – The size of historic time interval to iterate over at every batch model retrain. Leave at the default 0 to retrain on everything.
- **online_learning_rate** (*double*) – The learning rate used in the online stochastic gradient descent updates.
- **online_regularization_rate** (*double*) – The coefficient for the L2 regularization term for online update.
- **online_negative_rate** (*int*) – The number of negative samples generated after online each update. Useful for implicit recommendation.

alpenglow.experiments.BatchFactorExperiment module

class alpenglow.experiments.BatchFactorExperiment.**BatchFactorExperiment**(*dimension=10*, *begin_min=-0.01*, *begin_max=0.01*, *learning_rate=0.05*, *regularization_rate=0.0*, *negative_rate=0.0*, *number_of_iterations=3*, *period_length=86400*, *timeframe_length=0*, *clear_model=False*)

Bases: *alpenglow.OnlineExperiment.OnlineExperiment*

Batch version of *alpenglow.experiments.FactorExperiment.FactorExperiment*, meaning it retrains its model periodically nd evaluates the latest model between two training points in an online fashion.

Parameters

- **dimension** (*int*) – The latent factor dimension of the factormodel.
- **begin_min** (*double*) – The factors are initialized randomly, sampling each element uniformly from the interval (begin_min, begin_max).
- **begin_max** (*double*) – See begin_min.
- **learning_rate** (*double*) – The learning rate used in the stochastic gradient descent updates.
- **regularization_rate** (*double*) – The coefficient for the L2 regularization term.
- **negative_rate** (*int*) – The number of negative samples generated after each update. Useful for implicit recommendation.
- **number_of_iterations** (*int*) – The number of iterations over the data in model retrain.
- **period_length** (*int*) – The amount of time between model retrains (seconds).
- **timeframe_length** (*int*) – The size of historic time interval to iterate over at every model retrain. Leave at the default 0 to retrain on everything.
- **clear_model** (*bool*) – Whether to clear the model between retrains.

alpenglow.experiments.ExternalModelExperiment module

class alpenglow.experiments.ExternalModelExperiment.**ExternalModelExperiment**(*period_length=86400*, *timeframe_length=0*, *period_mode="time"*)

Bases: *alpenglow.OnlineExperiment.OnlineExperiment*

Parameters

- **period_length** (*int*) – The period length in seconds (or samples, see period_mode).
- **timeframe_length** (*int*) – The size of historic time interval to iterate over at every batch model retrain. Leave at the default 0 to retrain on everything.
- **period_mode** (*string*) – Either "time" or "samplenum", the unit of period_length and timeframe_length.

alpenglow.experiments.FactorExperiment module

class alpenglow.experiments.FactorExperiment.**FactorExperiment**(*dimension=10*, *begin_min=-0.01*, *begin_max=0.01*, *learning_rate=0.05*, *regularization_rate=0.0*, *negative_rate=0.0*)

Bases: *alpenglow.OnlineExperiment.OnlineExperiment*

This class implements an online version of the well-known matrix factorization recommendation model [Koren2009] and trains it via stochastic gradient descent. The model is able to train on implicit data using negative sample generation, see [X.He2016] and the **negative_rate** parameter.

Parameters

- **dimension** (*int*) – The latent factor dimension of the factormodel.
- **begin_min** (*double*) – The factors are initialized randomly, sampling each element uniformly from the interval (begin_min, begin_max).
- **begin_max** (*double*) – See begin_min.
- **learning_rate** (*double*) – The learning rate used in the stochastic gradient descent updates.
- **regularization_rate** (*double*) – The coefficient for the L2 regularization term.
- **negative_rate** (*int*) – The number of negative samples generated after each update. Useful for implicit recommendation.

alpenglow.experiments.FmExperiment module

class alpenglow.experiments.FmExperiment.**FmExperiment**(*dimension=10*, *begin_min=-0.01*, *begin_max=0.01*, *learning_rate=0.05*, *negative_rate=0.0*, *user_attributes=None*, *item_attributes=None*)

Bases: *alpenglow.OnlineExperiment.OnlineExperiment*

This class implements an online version of the factorization machine algorithm [Rendle2012] and trains it via stochastic gradient descent. The model is able to train on implicit data using negative sample generation, see [X.He2016] and the **negative_rate** parameter. Note that interactions between separate attributes of a user and between separate attributes of an item are not modeled.

The item and user attributes can be provided through the **user_attributes** and **item_attributes** parameters. These each expect a file path pointing to the attribute files. The required format is similar to the one used by libfm: the i. line describes the attributes of user **i** in a space sepaterated list of **index:value** pairs. For example the line "3:1 10:0.5" as the first line of the file indicates that user 0 has 1 as the value of attribute 3, and 0.5 as the value of attribute 10. If the files are omitted, an identity matrix is assumed.

Notice: once an attribute file is provided, the identity matrix is no longer assumed. If you wish to have a separate latent vector for each id, you must explicitly provide the identity matrix in the attribute file itself.

Parameters

- **dimension** (*int*) – The latent factor dimension of the factormodel.
- **begin_min** (*double*) – The factors are initialized randomly, sampling each element uniformly from the interval (begin_min, begin_max).
- **begin_max** (*double*) – See begin_min.
- **learning_rate** (*double*) – The learning rate used in the stochastic gradient descent updates.
- **negative_rate** (*int*) – The number of negative samples generated after each update. Useful for implicit recommendation.
- **user_attributes** (*string*) – The file containing the user attributes, in the format described in the model description. Set None for no attributes (identity matrix).
- **item_attributes** (*string*) – The file containing the item attributes, in the format described in the model description. Set None for no attributes (identity matrix).

alpenglow.experiments.NearestNeighborExperiment module

class alpenglow.experiments.NearestNeighborExperiment.**NearestNeighborExperiment**(*gamma=0.8*, *direction="forward"*, *gamma_thresho*, *num_of_neighb*

Bases: *alpenglow.OnlineExperiment.OnlineExperiment*

This class implements an online version of a similarity based recommendation model. One of the earliest and most popular collaborative filtering algorithms in practice is the item-based nearest neighbor [Sarwar2001] For these algorithms similarity scores are computed between item pairs based on the co-occurrence of the pairs in the preference of users. Non-stationarity of the data can be accounted for e.g. with the introduction of a time-decay [Ding2005] .

Describing the algorithm more formally, let us denote by U_i the set of users that visited item i, by I_u the set of items visited by user u, and by s_{ui} the index of item i in the sequence of interactions of user u. The frequency based time-weighted similarity function is defined by $sim(j,i) = \frac{\sum_{u \in U_j \cap U_i} f(s_{ui} - s_{uj})}{|U_j|}$, where $f(\tau) = \gamma^\tau$ is the time decaying function. For non-stationary data we sum only over users that visit item j before item i, setting $f(\tau) = 0$ if $\tau < 0$. For stationary data the absolute value of τ is used. The score assigned to item i for user u is $score(u,i) = \sum_{j \in I_u} f(|I_u| - s_{uj})\, sim(j,i)$. The model is represented by the similarity scores. Since computing the model is time consuming, it is done periodically. Moreover, only the most similar items are stored for each item. When the prediction scores are computed for a particular user, all items visited by the user can be considered, including the most recent ones. Hence, the algorithm can be considered semi-online in that it uses the most recent interactions of the current user, but not of the other users. We note that the time decay function is used here to quantify the strength of connection between pairs of items depending on how closely are located in the sequence of a user, and not as a way to forget old data as in [Ding2005].

Parameters

- **gamma** (*double*) – The constant used in the decay function. It shoud be set to 1 in offline and stationary experiments.
- **direction** (*string*) – Set to "forward" to consider the order of item pairs. Set to "both" when the order is not relevant.
- **gamma_thresold** (*double*) – Threshold to omit very small members when summing similarity. If the value of the decay function is smaller than the threshold, we omit the following members. Defaults to 0 (do not omit small members).
- **num_of_neighbors** (*int*) – The number of most similar items that will be stored in the model.

alpenglow.experiments.OldFactorExperiment module

class alpenglow.experiments.OldFactorExperiment.**OldFactorExperiment**(*dimension=10*, *begin_min=-0.01*, *begin_max=0.01*, *learning_rate=0.05*, *regularization_rate=0.0*, *negative_rate=0.0*)

Bases: *alpenglow.OnlineExperiment.OnlineExperiment*

This class implements an online version of the well-known matrix factorization recommendation model [Koren2009] and trains it via stochastic gradient descent. The model is able to train on implicit data using negative sample generation, see [X.He2016] and the **negative_rate** parameter.

Parameters

- **dimension** (*int*) – The latent factor dimension of the factormodel.
- **begin_min** (*double*) – The factors are initialized randomly, sampling each element uniformly from the interval (begin_min, begin_max).
- **begin_max** (*double*) – See begin_min.
- **learning_rate** (*double*) – The learning rate used in the stochastic gradient descent updates.

- **regularization_rate** (*double*) – The coefficient for the L2 regularization term.
- **negative_rate** (*int*) – The number of negative samples generated after each update. Useful for implicit recommendation.

alpenglow.experiments.PersonalPopularityExperiment module

class alpenglow.experiments.PersonalPopularityExperiment.**PersonalPopularityExperiment**(***para*
Bases: *alpenglow.OnlineExperiment.OnlineExperiment*

Recommends the item that the user has watched the most so far; in case of a tie, it falls back to global popularity. Running this model in conjunction with **exclude_known** == True is not recommended.

alpenglow.experiments.PopularityExperiment module

class alpenglow.experiments.PopularityExperiment.**PopularityExperiment**(***parameters*)
Bases: *alpenglow.OnlineExperiment.OnlineExperiment*

Recommends the most popular item from the set of items seen so far.

alpenglow.experiments.PopularityTimeframeExperiment module

class alpenglow.experiments.PopularityTimeframeExperiment.**PopularityTimeframeExperiment**(*tau*
Bases: *alpenglow.OnlineExperiment.OnlineExperiment*

Time-aware version of PopularityModel, which only considers the last **tau** time interval when calculating popularities.

Parameters **tau** (*int*) – The time amount to consider.

alpenglow.experiments.SvdppExperiment module

class alpenglow.experiments.SvdppExperiment.**SvdppExperiment**(*begin_min=-0.01*, *begin_max=0.01*, *dimension=10*, *use_sigmoid=False*, *norm_type="exponential"*, *gamma=0.8*, *user_vector_weight=0.5*, *history_weight=0.5*)
Bases: *alpenglow.OnlineExperiment.OnlineExperiment*

This class implements an online version of the SVD++ model [Koren2008] The model is able to train on implicit data using negative sample generation, see [X.He2016] and the **negative_rate** parameter. We apply a decay on the user history, the weight of the older items is smaller.

Parameters

- **begin_min** (*double*) – The factors are initialized randomly, sampling each element uniformly from the interval (begin_min, begin_max).
- **begin_max** (*double*) – See begin_min.
- **dimension** (*int*) – The latent factor dimension of the factormodel.
- **learning_rate** (*double*) – The learning rate used in the stochastic gradient descent updates.

- **negative_rate** (*int*) – The number of negative samples generated after each update. Useful for implicit recommendation.
- **norm_type** (*string*) – Normalization variants.
- **gamma** (*double*) – The constant in the decay function.
- **user_vector_weight** (*double*) – The user is modeled with a sum of a user vector and a combination of item vectors. The weight of the two part can be set using these parameters.
- **history_weight** (*double*) – See user_vector_weight.

alpenglow.experiments.TransitionProbabilityExperiment module

class alpenglow.experiments.TransitionProbabilityExperiment.**TransitionProbabilityExperiment**
Bases: *alpenglow.OnlineExperiment.OnlineExperiment*

A simple algorithm that focuses on the sequence of items a user has visited is one that records how often users visited item i after visiting another item j. This can be viewed as particular form of the item-to-item nearest neighbor with a time decay function that is non-zero only for the immediately preceding item. While the algorithm is more simplistic, it is fast to update the transition fre- quencies after each interaction, thus all recent information is taken into account.

Parameters **mode** (*string*) – The direction of transitions to be considered.

Module contents

6.1.3 alpenglow.offline package

Subpackages

alpenglow.offline.evaluation package

Submodules

alpenglow.offline.evaluation.NdcgScore module

alpenglow.offline.evaluation.NdcgScore.**NdcgScore**(*test*, *recommendations*, *top_k=100*)

alpenglow.offline.evaluation.PrecisionScore module

alpenglow.offline.evaluation.PrecisionScore.**PrecisionScore**(*test*, *recommendations*, *top_k*)

alpenglow.offline.evaluation.RecallScore module

alpenglow.offline.evaluation.RecallScore.**RecallScore**(*test*, *recommendations*, *top_k*)

Module contents

alpenglow.offline.models package

Submodules

alpenglow.offline.models.ALSFactorModel module

class alpenglow.offline.models.ALSFactorModel.**ALSFactorModel**(*dimension=10*, *begin_min=-0.01*, *begin_max=0.01*, *number_of_iterations=3*, *regularization_lambda=0.0001*, *alpha=40*, *implicit=1*)

Bases: *alpenglow.offline.OfflineModel.OfflineModel*

This class implements the well-known matrix factorization recommendation model [Koren2009] and trains it using ALS and iALS [Hu2008].

Parameters

- **dimension** (*int*) – The latent factor dimension of the factormodel.
- **begin_min** (*double*) – The factors are initialized randomly, sampling each element uniformly from the interval (begin_min, begin_max).
- **begin_max** (*double*) – See begin_min.
- **number_of_iterations** (*double*) – Number of times to optimize the user and the item factors for least squares.
- **regularization_lambda** (*double*) – The coefficient for the L2 regularization term. See [Hu2008]. This number is multiplied by the number of non-zero elements of the user-item rating matrix before being used, to achieve similar magnitude to the one used in traditional SGD.
- **alpha** (*int*) – The weight coefficient for positive samples in the error formula in the case of implicit factorization. See [Hu2008].
- **implicit** (*int*) – Whether to treat the data as implicit (and optimize using iALS) or explicit (and optimize using ALS).

alpenglow.offline.models.AsymmetricFactorModel module

class alpenglow.offline.models.AsymmetricFactorModel.**AsymmetricFactorModel**(*dimension=10*, *begin_min=-0.01*, *begin_max=0.01*, *learning_rate=0.05*, *regularization_rate=0.0*, *negative_rate=0*, *number_of_iterations=9*)

Bases: *alpenglow.offline.OfflineModel.OfflineModel*

Implements the recommendation model introduced in [Paterek2007].

Parameters

- **dimension** (*int*) – The latent factor dimension of the factormodel.
- **begin_min** (*double*) – The factors are initialized randomly, sampling each element uniformly from the interval (begin_min, begin_max).
- **begin_max** (*double*) – See begin_min.
- **learning_rate** (*double*) – The learning rate used in the stochastic gradient descent updates.
- **regularization_rate** (*double*) – The coefficient for the L2 regularization term.
- **negative_rate** (*int*) – The number of negative samples generated after each update. Useful for implicit recommendation.
- **number_of_iterations** (*int*) – Number of times to iterate over the training data.

alpenglow.offline.models.FactorModel module

class alpenglow.offline.models.FactorModel.**FactorModel**(*dimension=10*, *begin_min=-0.01*, *begin_max=0.01*, *learning_rate=0.05*, *regularization_rate=0.0*, *negative_rate=0.0*, *number_of_iterations=9*)

Bases: *alpenglow.offline.OfflineModel.OfflineModel*

This class implements the well-known matrix factorization recommendation model [Koren2009] and trains it via stochastic gradient descent. The model is able to train on implicit data using negative sample generation, see [X.He2016] and the **negative_rate** parameter.

Parameters

- **dimension** (*int*) – The latent factor dimension of the factormodel.

- **begin_min** (*double*) – The factors are initialized randomly, sampling each element uniformly from the interval (begin_min, begin_max).
- **begin_max** (*double*) – See begin_min.
- **learning_rate** (*double*) – The learning rate used in the stochastic gradient descent updates.
- **regularization_rate** (*double*) – The coefficient for the L2 regularization term.
- **negative_rate** (*int*) – The number of negative samples generated after each update. Useful for implicit recommendation.
- **number_of_iterations** (*int*) – Number of times to iterate over the training data.

alpenglow.offline.models.NearestNeighborModel module

class alpenglow.offline.models.NearestNeighborModel.**NearestNeighborModel**(*num_of_neighbors=10*)

Bases: *alpenglow.offline.OfflineModel.OfflineModel*

One of the earliest and most popular collaborative filtering algorithms in practice is the item-based nearest neighbor [Sarwar2001] For these algorithms similarity scores are computed between item pairs based on the co-occurrence of the pairs in the preference of users. Non-stationarity of the data can be accounted for e.g. with the introduction of a time-decay [Ding2005] .

Describing the algorithm more formally, let us denote by U_i the set of users that visited item i, by I_u the set of items visited by user u, and by s_{ui} the index of item i in the sequence of interactions of user u. The frequency based similarity function is defined by $sim(j,i) = \frac{\sum_{u \in U_j \cap U_i} 1}{|U_j|}$. The score assigned to item i for user u is $score(u,i) = \sum_{j \in I_u} sim(j,i)$. The model is represented by the similarity scores. Only the most similar items are stored for each item. When the prediction scores are computed for a particular user, all items visited by the user are considered.

Parameters **num_of_neighbors** (*int*) – Number of most similar items that will be stored in the model.

alpenglow.offline.models.PopularityModel module

class alpenglow.offline.models.PopularityModel.**PopularityModel**

Bases: *alpenglow.offline.OfflineModel.OfflineModel*

Recommends the most popular item from the set of items.

alpenglow.offline.models.SvdppModel module

class alpenglow.offline.models.SvdppModel.**SvdppModel**(*dimension=10*, *begin_min=-0.01*, *begin_max=0.01*, *learning_rate=0.05*, *negative_rate=0.0*, *number_of_iterations=20*, *cumulative_item_updates=false*)

Bases: *alpenglow.offline.OfflineModel.OfflineModel*

This class implements the SVD++ model [Koren2008] The model is able to train on implicit data using negative sample generation, see [X.He2016] and the **negative_rate** parameter.

Parameters

- **dimension** (*int*) – The latent factor dimension of the factormodel.
- **begin_min** (*double*) – The factors are initialized randomly, sampling each element uniformly from the interval (begin_min, begin_max).
- **begin_max** (*double*) – See begin_min.
- **learning_rate** (*double*) – The learning rate used in the stochastic gradient descent updates.
- **negative_rate** (*int*) – The number of negative samples generated after each update. Useful for implicit recommendation.
- **number_of_iterations** (*int*) – Number of times to iterate over the training data.
- **cumulative_item_updates** (*boolean*) – Cumulative item updates make the model faster but less accurate.

Module contents

Submodules

alpenglow.offline.OfflineModel module

class alpenglow.offline.OfflineModel.**OfflineModel**(***parameters*)

Bases: *alpenglow.ParameterDefaults.ParameterDefaults*

OfflineModel is the base class for all traditional, scikit-learn style models in Alpenglow. Example usage:

```
data = pd.read_csv('data')
train_data = data[data.time < (data.time.min()+250*86400)]
test_data = data[ (data.time >= (data.time.min()+250*86400)) & (data.time < (data.
↪time.min()+300*86400))]

exp = ag.offline.models.FactorModel(
    learning_rate=0.07,
    negative_rate=70,
    number_of_iterations=9,
)
exp.fit(data)
test_users = list(set(test_data.user)&set(train_data.user))
recommendations = exp.recommend(users=test_users)
```

fit(*X*, *y=None*, *columns={}*)

Fit the model to a dataset.

Parameters

- **X** (*pandas.DataFrame*) – The input data, must contain the columns **user** and **item**. May contain the **score** column as well.
- **y** (*pandas.Series or list*) – The target values. If not set (and X doesn't contain the score column), it is assumed to be constant 1 (implicit recommendation).
- **columns** (*dict*) – Optionally the mapping of the input DataFrame's columns' names to the expected ones.

predict(*X*)

Predict the target values on X.

Parameters **X** (*pandas.DataFrame*) – The input data, must contain the columns **user** and **item**.

Returns List of predictions

Return type list

recommend(*users=None*, *k=100*, *exclude_known=True*)
Give toplist recommendations for users.

Parameters

- **users** (*list*) – List of users to give recommendation for.
- **k** (*int*) – Size of toplists
- **exclude_known** (*bool*) – Whether to exclude (user,item) pairs in the train dataset from the toplists.

Returns DataFrame of recommendations, with columns **user**, **item** and **rank**.

Return type pandas.DataFrame

Module contents

6.1.4 alpenglow.utils package

Submodules

alpenglow.utils.AvailabilityFilter module

class alpenglow.utils.AvailabilityFilter.**AvailabilityFilter**(*availability_data*)
Bases: *alpenglow.cpp.AvailabilityFilter*

Python wrapper around *alpenglow.cpp.AvailabilityFilter*.

alpenglow.utils.DataframeData module

class alpenglow.utils.DataframeData.**DataframeData**(*df*, *columns={}*)
Bases: *alpenglow.cpp.DataframeData*

Python wrapper around *alpenglow.cpp.DataframeData*.

alpenglow.utils.FactorModelReader module

alpenglow.utils.FactorModelReader.**readEigenFactorModel**(*file*)

alpenglow.utils.FactorModelReader.**readFactorModel**(*file*, *dimensions*)

alpenglow.utils.ParameterSearch module

class alpenglow.utils.ParameterSearch.**DependentParameter**(*format_string*, *parameter_names=None*)
Bases: object

eval(*parameters*)

class alpenglow.utils.ParameterSearch.**ParameterSearch**(*model*, *Score*)
Bases: object

Utility for evaluating online experiments with different hyperparameters. For a brief tutorial on using this class, see *Five minute tutorial*.

run(**run_paremeters*, ***run_kw_parameters*)

set_parameter_values(*parameter_name*, *parameter_values*)

alpenglow.utils.ThreadedParameterSearch module

class alpenglow.utils.ThreadedParameterSearch.**ThreadedParameterSearch**(*model*, *Score*, *threads=4*, *use_process_pool=True*)
Bases: *alpenglow.utils.ParameterSearch.ParameterSearch*

Threaded version of *alpenglow.utils.ParameterSearch*.

run(**run_parameters*, ***run_kw_parameters*)

Module contents

6.2 Submodules

6.3 alpenglow.Getter module

class alpenglow.Getter.**Getter**
Bases: object

Responsible for creating and managing cpp objects in the alpenglow.cpp package.

collect_ = {}

items = {}

class alpenglow.Getter.**MetaGetter**(*a*, *b*, *c*)
Bases: type

Metaclass of *alpenglow.Getter.Getter*. Provides utilities for creating and managing cpp objects in the alpenglow.cpp package. For more information, see /general/memory_management.

collect()

get_and_clean()

initialize_all(*objects*)

run_self_test(*i*)

set_experiment_environment(*online_experiment*, *objects*)

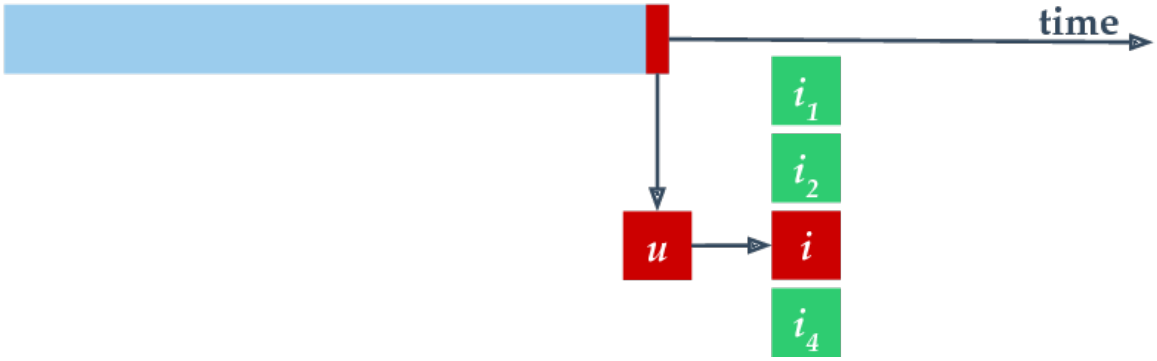
6.4 alpenglow.OnlineExperiment module

class alpenglow.OnlineExperiment.**OnlineExperiment**(*seed=254938879*, *top_k=100*)
Bases: *alpenglow.ParameterDefaults.ParameterDefaults*

This is the base class of every online experiment in Alpenglow. It builds the general experimental setup needed to run the online training and evaluation of a model. It also handles default parameters and the ability to override them when instantiating an experiment.

Subclasses should implement the `config()` method; for more information, check the documentation of this method as well.

Online evaluation in Alpenglow is done by processing the data row-by-row and evaluating the model on each new record before providing the model with the new information.



Evaluation is done by ranking the next item on the user's toplist and saving the rank. If the item is not found in the top `top_k` items, the evaluation step returns `NaN`.

For a brief tutorial on using this class, see *Five minute tutorial*.

Parameters

- **seed** (*int*) – The seed to initialize RNG-s. Should not be 0.
- **top_k** (*int*) – The length of the toplists.

get_predictions()

If the `calculate_toplists` parameter is set when calling `run`, this method can used to acquire the generated toplists.

Returns

DataFrame containing the columns record_id, time, user, item, rank and prediction.

- **record_id** is the index of the record begin evaluated in the input DataFrame. Generally, there are `top_k` rows with the same record_id.
- **time** is the time of the evaluation
- **user** is the user the toplist is generated for
- **item** is the item of the toplist at the **rank** place
- **prediction** is the prediction given by the model for the (user, item) pair at the time of evaluation.

Return type pandas.DataFrame

run(*data*, *experimentType=None*, *columns={}*, *verbose=True*, *out_file=None*, *exclude_known=False*, *initialize_all=False*, *max_item=-1*, *max_user=-1*, *calculate_toplists=False*, *max_time=0*, *memory_log=True*, *shuffle_same_time=True*)

Parameters

- **data** (*pandas.DataFrame or str*) – The input data, see *Five minute tutorial*. If this parameter is a string, it has to be in the format specified by `experimentType`.

- **experimentType** (*str*) – The format of the input file if `data` is a string
- **columns** (*dict*) – Optionally the mapping of the input DataFrame's columns' names to the expected ones.
- **verbose** (*bool*) – Whether to write information about the experiment while running
- **out_file** (*str*) – If set, the results of the experiment are also written to the file located at `out_file`.
- **exclude_known** (*bool*) – If set to True, a user's previosly seen items are excluded from the toplist evaluation. The `eval` columns of the input data should be set accordingly.
- **calculate_toplists** (*bool or list*) – Whether to actually compute the toplists or just the ranks (the latter is faster). It can be specified on a record-by-record basis, by giving a list of booleans as parameter. The calculated toplists can be acquired after the experiment's end by using `get_predictions`. Setting this to non-False implies shuffle_same_time=False
- **max_time** (*int*) – Stop the experiment at this timestamp.
- **memory_log** (*bool*) – Whether to log the results to memory (to be used optionally with out_file)
- **shuffle_same_time** (*bool*) – Whether to shuffle records with the same timestamp randomly.

Returns Results DataFrame if memory_log=True, empty DataFrame otherwise

Return type DataFrame

6.5 alpenglow.ParameterDefaults module

class alpenglow.ParameterDefaults.**ParameterDefaults**(***parameters*)

Bases: `object`

Base class of OnlineExperiment and OfflineModel, providing utilities for parameter defaults and overriding.

check_unused_parameters()

parameter_default(*name*, *value*)

parameter_defaults(***defaults*)

set_parameter(*name*, *value*)

6.6 Module contents

CHAPTER 7

alpenglow.cpp package

The classes in this module are usually not used directly, but instead through the *alpenglow.Getter* class. For more info, read TODO: named parameters, memory management and self_test().

7.1 loggers

class alpenglow.cpp.**InputLoggerParameters**
Bases: sip.wrapper

output_file

class alpenglow.cpp.**InputLogger**
Bases: *alpenglow.cpp.Logger*, *alpenglow.cpp.Initializable*

autocalled_initialize()

run()

self_test()

class alpenglow.cpp.**Logger**
Bases: sip.wrapper

run()

self_test()

class alpenglow.cpp.**RankingLog**
Bases: sip.wrapper

id

item

prediction

rank

score

time

user

class alpenglow.cpp.**RankingLogs**
Bases: sip.wrapper

logs

top_k

class alpenglow.cpp.**MemoryRankingLoggerParameters**
Bases: sip.wrapper

memory_log

min_time

out_file

class alpenglow.cpp.**MemoryRankingLogger**
Bases: *alpenglow.cpp.Logger*

run()

set_model()

set_rank_computer()

set_ranking_logs()

class alpenglow.cpp.**OnlinePredictorParameters**
Bases: sip.wrapper

file_name

min_time

time_frame

class alpenglow.cpp.**OnlinePredictor**
Bases: *alpenglow.cpp.Logger*

run()

self_test()

set_prediction_creator()

class alpenglow.cpp.**OnlinePredictions**
Bases: sip.wrapper

ids

items

ranks

scores

times

users

class alpenglow.cpp.**PredictionLogger**
Bases: *alpenglow.cpp.Logger*

get_predictions()

run()

self_test()

set_prediction_creator()

class alpenglow.cpp.**InterruptLogger**
Bases: *alpenglow.cpp.Logger*

run()

class alpenglow.cpp.**ListConditionalMetaLoggerParameters**
Bases: sip.wrapper

should_run_vector

class alpenglow.cpp.**ListConditionalMetaLogger**
Bases: *alpenglow.cpp.ConditionalMetaLogger*

should_run()

class alpenglow.cpp.**ConditionalMetaLogger**
Bases: *alpenglow.cpp.Logger*

run()

self_test()

set_logger()

should_run()

class alpenglow.cpp.**ProceedingLogger**
Bases: *alpenglow.cpp.Logger*, *alpenglow.cpp.Initializable*, *alpenglow.cpp.NeedsExperimentEnvironment*

autocalled_initialize()

run()

self_test()

set_data_iterator()

set_experiment_environment()

7.2 online_experiment

class alpenglow.cpp.**OnlineExperimentParameters**
Bases: sip.wrapper

exclude_known

initialize_all

max_item

max_time

max_user

min_time

random_seed

top_k

class alpenglow.cpp.**OnlineExperiment**
Bases: sip.wrapper

add_logger()

add_updater()

inject_experiment_environment_into()

run()

self_test()

set_recommender_data_iterator()

class alpenglow.cpp.**ExperimentEnvironment**
Bases: sip.wrapper

do_exclude_known()

get_max_time()

get_min_time()

get_popularity_container()

get_popularity_sorted_container()

get_random()

get_recommender_data_iterator()

get_top_k()

get_train_matrix()

is_item_new_for_user()

set_parameters()

update()

7.3 data_generators

class alpenglow.cpp.**CompletePastDataGenerator**
Bases: *alpenglow.cpp.DataGenerator*, *alpenglow.cpp.NeedsExperimentEnvironment*, *alpenglow.cpp.Initializable*

autocalled_initialize()

generate_recommender_data()

self_test()

set_experiment_environment()

set_recommender_data_iterator()

class alpenglow.cpp.**SamplingDataGeneratorParameters**
Bases: sip.wrapper

distribution

geometric_param

number_of_samples

y

class alpenglow.cpp.**SamplingDataGenerator**
Bases: *alpenglow.cpp.DataGenerator*, *alpenglow.cpp.Initializable*, *alpenglow.cpp.NeedsExperimentEnvironment*

autocalled_initialize()

generate_recommender_data()

self_test()

set_experiment_environment()

set_recommender_data_iterator()

class alpenglow.cpp.**TimeframeDataGeneratorParameters**
Bases: sip.wrapper

timeframe_length

class alpenglow.cpp.**TimeframeDataGenerator**
Bases: *alpenglow.cpp.DataGenerator*, *alpenglow.cpp.NeedsExperimentEnvironment*, *alpenglow.cpp.Initializable*

autocalled_initialize()

generate_recommender_data()

self_test()

set_experiment_environment()

set_recommender_data_iterator()

class alpenglow.cpp.**DataGenerator**
Bases: sip.wrapper

generate_recommender_data()

7.4 online_learners

class alpenglow.cpp.**PeriodicOfflineLearnerWrapperParameters**
Bases: sip.wrapper

base_in_file_name

base_out_file_name

clear_model

learn

read_model

write_model

class alpenglow.cpp.**PeriodicOfflineLearnerWrapper**
Bases: *alpenglow.cpp.Updater*

add_offline_learner()

self_test()

set_data_generator()

set_model()

set_period_computer()

update()

class alpenglow.cpp.**LearnerPeriodicDelayedWrapperParameters**
Bases: sip.wrapper

delay

period

class alpenglow.cpp.**LearnerPeriodicDelayedWrapper**
Bases: *alpenglow.cpp.Updater*

self_test()

set_wrapped_learner()

update()

7.5 general_interfaces

class alpenglow.cpp.**Initializable**
Bases: sip.wrapper

This interface signals that the implementing class has to be initialized by the experiment runner. The experiment runner calls the *initialize()* method, which in return calls the class-specific implementation of *autocalled_initialize()* and sets the *is_initialized()* flag if the initialization was successful. The *autocalled_initialize()* method can check whether the neccessary dependencies have been initialized or not before initializing the instance; and should return the success value accordingly.

If the initialization was not successful, the experiment runner keeps trying to initialize the not-yet initialized objects, thus resolving dependency chains.

Initializing and inheritance. Assume that class Parent implements Initializable, and the descendant Child needs further initialization. In that case Child has to override *autocalled_initialize()*, and call *Parent::autocalled_initialize()* in the overriding function first, continuing only if the parent returned true. If the init of the parent was succesful, but the children failed, then the children has to store the success of the parent and omit calling the initialization of the parent later.

autocalled_initialize()
Has to be implemented by the component.

Returns Whether the initialization was successful.

Return type bool

initialize()

Returns Whether the initialization was successful.

Return type bool

is_initialized()

Returns Whether the component has already been initialized.

Return type bool

class alpenglow.cpp.**NeedsExperimentEnvironment**
Bases: sip.wrapper

set_experiment_environment()

class alpenglow.cpp.**Updater**
Bases: sip.wrapper

self_test()

update()

7.6 objectives

class alpenglow.cpp.**ObjectivePointWise**
Bases: sip.wrapper

get_gradient()

class alpenglow.cpp.**ObjectivePairWise**
Bases: sip.wrapper

class alpenglow.cpp.**ObjectiveListWise**
Bases: sip.wrapper

get_gradient()

class alpenglow.cpp.**ObjectiveMSE**
Bases: *alpenglow.cpp.ObjectivePointWise*

get_gradient()

7.7 negative_sample_generators

class alpenglow.cpp.**UniformNegativeSampleGeneratorParameters**
Bases: sip.wrapper

filter_repeats

initialize_all

max_item

negative_rate

seed

class alpenglow.cpp.**UniformNegativeSampleGenerator**
Bases: *alpenglow.cpp.NegativeSampleGenerator*, *alpenglow.cpp.Initializable*, *alpenglow.cpp.NeedsExperimentEnvironment*

autocalled_initialize()

self_test()

set_experiment_environment()

set_items()

set_train_matrix()

class alpenglow.cpp.**NegativeSampleGenerator**
Bases: *alpenglow.cpp.Updater*

add_updater()

self_test()

update()

7.8 offline_evaluators

class alpenglow.cpp.**OfflineEvaluator**
Bases: sip.wrapper

evaluate()

self_test()

class alpenglow.cpp.**PrecisionRecallEvaluatorParameters**
Bases: sip.wrapper

cutoff

test_file_name

test_file_type

time

class alpenglow.cpp.**PrecisionRecallEvaluator**
Bases: *alpenglow.cpp.OfflineEvaluator*

evaluate()

self_test()

set_model()

set_model_filter()

set_train_data()

class alpenglow.cpp.**OfflineRankingComputerParameters**
Bases: sip.wrapper

top_k

class alpenglow.cpp.**OfflinePredictions**
Bases: sip.wrapper

items

ranks

users

class alpenglow.cpp.**OfflineRankingComputer**
Bases: sip.wrapper

compute()

set_items()

set_toplist_creator()

set_users()

7.9 utils

class alpenglow.cpp.**Random**
Bases: sip.wrapper

get()

get_arctg()

get_boolean()

get_discrete()

get_geometric()

get_linear()

set()

class alpenglow.cpp.**PopContainer**
Bases: sip.wrapper

clear()

get()

increase()

reduce()

resize()

class alpenglow.cpp.**TopPopContainer**
Bases: sip.wrapper

get_index()

get_item()

has_changed()

increase()

reduce()

set_threshold()

size()

class alpenglow.cpp.**SpMatrix**
Bases: sip.wrapper

clear()

erase()

get()

has_value()

increase()

insert()

read_from_file()

resize()

row_size()

size()

update()

write_into_file()

class alpenglow.cpp.**Bias**
Bases: sip.wrapper

clear()

get()

init()

update()

class alpenglow.cpp.**SparseAttributeContainerParameters**
Bases: sip.wrapper

class alpenglow.cpp.**SparseAttributeContainer**
Bases: sip.wrapper

get_max_attribute_index()

class alpenglow.cpp.**FileSparseAttributeContainer**
Bases: *alpenglow.cpp.SparseAttributeContainer*

load_from_file()

class alpenglow.cpp.**PredictionCreatorParameters**
Bases: sip.wrapper

exclude_known

top_k

class alpenglow.cpp.**PredictionCreator**
Bases: *alpenglow.cpp.NeedsExperimentEnvironment*, *alpenglow.cpp.Initializable*

autocalled_initialize()

run()

self_test()

set_experiment_environment()

set_filter()

set_model()

set_train_matrix()

class alpenglow.cpp.**PredictionCreatorGlobalParameters**
Bases: *alpenglow.cpp.PredictionCreatorParameters*

initial_threshold

class alpenglow.cpp.**PredictionCreatorGlobal**
Bases: *alpenglow.cpp.PredictionCreator*

autocalled_initialize()

run()

self_test()

class alpenglow.cpp.**PredictionCreatorPersonalizedParameters**
Bases: *alpenglow.cpp.PredictionCreatorParameters*

class alpenglow.cpp.**PredictionCreatorPersonalized**
Bases: *alpenglow.cpp.PredictionCreator*

autocalled_initialize()

run()

self_test()

class alpenglow.cpp.**PeriodComputerParameters**
Bases: sip.wrapper

period_length

period_mode

start_time

class alpenglow.cpp.**PeriodComputer**
Bases: *alpenglow.cpp.Updater*, *alpenglow.cpp.NeedsExperimentEnvironment*, *alpenglow.cpp.Initializable*

autocalled_initialize()

end_of_period()

get_period_num()

self_test()

set_experiment_environment()

set_parameters()

set_recommender_data_iterator()

update()

class alpenglow.cpp.**Recency**
Bases: sip.wrapper

get()

update()

class alpenglow.cpp.**PowerLawRecencyParameters**
Bases: sip.wrapper

delta_t

exponent

class alpenglow.cpp.**PowerLawRecency**
Bases: *alpenglow.cpp.Recency*

get()

update()

7.10 gradient_computers

class alpenglow.cpp.**GradientComputer**
Bases: *alpenglow.cpp.Updater*

add_gradient_updater()

self_test()

set_model()

class alpenglow.cpp.**GradientComputerPointWise**
Bases: *alpenglow.cpp.GradientComputer*

self_test()

set_objective()

update()

7.11 recommender_data

class alpenglow.cpp.**InlineAttributeReader**
Bases: sip.wrapper

read_attribute()

self_test()

class alpenglow.cpp.**DataframeData**
Bases: *alpenglow.cpp.RecommenderData*

add_recdats()

autocalled_initialize()

get()

size()

class alpenglow.cpp.**ShuffleIteratorParameters**
Bases: sip.wrapper

seed

class alpenglow.cpp.**ShuffleIterator**
Bases: *alpenglow.cpp.RecommenderDataIterator*

autocalled_initialize()

get()

get_actual()

get_following_timestamp()

get_future()

next()

class alpenglow.cpp.**RandomIteratorParameters**
Bases: sip.wrapper

seed

shuffle_mode

class alpenglow.cpp.**RandomIterator**
Bases: *alpenglow.cpp.RecommenderDataIterator*

autocalled_initialize()

get()

get_actual()

get_following_timestamp()

get_future()

next()

restart()

shuffle()

class alpenglow.cpp.**RecDat**
Bases: sip.wrapper

category

eval

id

item

score

time

user

class alpenglow.cpp.**RecPred**
Bases: sip.wrapper

prediction

score

class alpenglow.cpp.**RecommenderData**
Bases: *alpenglow.cpp.Initializable*

autocalled_initialize()

clear()

get()

get_all_items()

get_all_users()

get_full_matrix()

get_items_into()

get_rec_data()

get_users_into()

set_rec_data()

size()

class alpenglow.cpp.**LegacyRecommenderDataParameters**
Bases: sip.wrapper

file_name

max_time

type

class alpenglow.cpp.**LegacyRecommenderData**
Bases: *alpenglow.cpp.RecommenderData*

autocalled_initialize()

read_from_file()

set_attribute_container()

class alpenglow.cpp.**FactorRepr**
Bases: sip.wrapper

entity

factors

class alpenglow.cpp.**UserItemFactors**
Bases: sip.wrapper

item_factors

user_factors

class alpenglow.cpp.**FactorModelReader**
Bases: sip.wrapper

read()

class alpenglow.cpp.**EigenFactorModelReader**
Bases: sip.wrapper

read()

class alpenglow.cpp.**SimpleIterator**
Bases: *alpenglow.cpp.RecommenderDataIterator*

autocalled_initialize()

get()

get_actual()

get_following_timestamp()

get_future()

next()

class alpenglow.cpp.**RecommenderDataIterator**
Bases: *alpenglow.cpp.Initializable*

autocalled_initialize()

get()

get_actual()

get_counter()

get_following_timestamp()

get_future()

has_next()

next()

set_recommender_data()

size()

7.12 models

7.12.1 models.baseline

class alpenglow.cpp.**PersonalPopularityModel**
Bases: *alpenglow.cpp.Model*

prediction()

class alpenglow.cpp.**TransitionProbabilityModelUpdaterParameters**
Bases: sip.wrapper

filter_freq_updates

label_file_name

label_transition_mode

mode

class alpenglow.cpp.**TransitionProbabilityModelUpdater**
Bases: *alpenglow.cpp.Updater*

self_test()

set_model()

update()

class alpenglow.cpp.**PopularityModelUpdater**
Bases: *alpenglow.cpp.Updater*

self_test()

set_model()

update()

class alpenglow.cpp.**PopularityModel**
Bases: *alpenglow.cpp.Model*

prediction()

class alpenglow.cpp.**PopularityTimeFrameModelUpdaterParameters**
Bases: sip.wrapper

tau

class alpenglow.cpp.**PopularityTimeFrameModelUpdater**
Bases: *alpenglow.cpp.Updater*

self_test()

set_model()

update()

class alpenglow.cpp.**NearestNeighborModelParameters**
Bases: sip.wrapper

direction

gamma

gamma_threshold

norm

num_of_neighbors

class alpenglow.cpp.**NearestNeighborModel**
Bases: *alpenglow.cpp.Model*

prediction()

self_test()

class alpenglow.cpp.**NearestNeighborModelUpdaterParameters**
Bases: sip.wrapper

compute_similarity_period

period_mode

class alpenglow.cpp.**NearestNeighborModelUpdater**
Bases: *alpenglow.cpp.Updater*

self_test()

set_model()

update()

class alpenglow.cpp.**PersonalPopularityModelUpdater**
Bases: *alpenglow.cpp.Updater*

self_test()

set_model()

update()

class alpenglow.cpp.**TransitionProbabilityModel**
Bases: *alpenglow.cpp.Model*

clear()

prediction()

self_test()

7.12.2 models.factor

class alpenglow.cpp.**FmModelParameters**
Bases: sip.wrapper

begin_max

begin_min

dimension

item_attributes

seed

user_attributes

class alpenglow.cpp.**FmModel**
Bases: *alpenglow.cpp.Model*, *alpenglow.cpp.Initializable*

autocalled_initialize()

clear()

prediction()

self_test()

class alpenglow.cpp.**SvdppModelParameters**
Bases: sip.wrapper

begin_max

begin_min

dimension

gamma

history_weight

norm_type

seed

use_sigmoid

user_vector_weight

class alpenglow.cpp.**SvdppModel**
Bases: *alpenglow.cpp.Model*

add()

clear()

prediction()

self_test()

class alpenglow.cpp.**SvdppModelUpdater**
Bases: *alpenglow.cpp.Updater*

self_test()

set_model()

update()

class alpenglow.cpp.**AsymmetricFactorModelGradientUpdaterParameters**
Bases: sip.wrapper

cumulative_item_updates

learning_rate

class alpenglow.cpp.**AsymmetricFactorModelGradientUpdater**
Bases: *alpenglow.cpp.ModelGradientUpdater*

beginning_of_updating_cycle()

end_of_updating_cycle()

self_test()

set_model()

update()

class alpenglow.cpp.**AsymmetricFactorModelParameters**
Bases: sip.wrapper

begin_max

begin_min

dimension

gamma

initialize_all

max_item

norm_type

seed

use_sigmoid

class alpenglow.cpp.**AsymmetricFactorModel**
Bases: *alpenglow.cpp.Model*

add()

clear()

prediction()

self_test()

class alpenglow.cpp.**FactorModelParameters**
Bases: sip.wrapper

begin_max

begin_min

dimension

initialize_all

max_item

max_user

use_item_bias

use_sigmoid

use_user_bias

class alpenglow.cpp.**FactorModel**
Bases: *alpenglow.cpp.Model*, *alpenglow.cpp.SimilarityModel*, *alpenglow.cpp.Initializable*

add()

autocalled_initialize()

clear()

prediction()

self_test()

set_item_recency()

set_user_recency()

similarity()

class alpenglow.cpp.**FactorModelGradientUpdaterParameters**
Bases: sip.wrapper

learning_rate

learning_rate_bias

regularization_rate

regularization_rate_bias

turn_off_item_bias_updates

turn_off_item_factor_updates

turn_off_user_bias_updates

turn_off_user_factor_updates

class alpenglow.cpp.**FactorModelGradientUpdater**
Bases: *alpenglow.cpp.ModelGradientUpdater*

self_test()

set_model()

update()

class alpenglow.cpp.**SvdppModelGradientUpdaterParameters**
Bases: sip.wrapper

cumulative_item_updates

learning_rate

class alpenglow.cpp.**SvdppModelGradientUpdater**
Bases: *alpenglow.cpp.ModelGradientUpdater*

beginning_of_updating_cycle()

end_of_updating_cycle()

self_test()

set_model()

update()

class alpenglow.cpp.**AsymmetricFactorModelUpdater**
Bases: *alpenglow.cpp.Updater*

self_test()

set_model()

update()

class alpenglow.cpp.**FmModelUpdaterParameters**
Bases: sip.wrapper

learning_rate

class alpenglow.cpp.**FmModelUpdater**
Bases: *alpenglow.cpp.Updater*

self_test()

set_model()

update()

class alpenglow.cpp.**EigenFactorModelParameters**
Bases: sip.wrapper

begin_max

begin_min

dimension

lemp_bucket_size

seed

class alpenglow.cpp.**EigenFactorModel**
Bases: *alpenglow.cpp.Model*, *alpenglow.cpp.Initializable*

add()

autocalled_initialize()

clear()

prediction()

resize()

self_test()

7.12.3 models.combination

class alpenglow.cpp.**WeightedModelStructure**
Bases: sip.wrapper

distribution_

is_initialized()

models_

class alpenglow.cpp.**WMSUpdater**
Bases: sip.wrapper

set_wms()

class alpenglow.cpp.**ToplistCombinationModel**
Bases: *alpenglow.cpp.Model*, *alpenglow.cpp.Initializable*, *alpenglow.cpp.NeedsExperimentEnvironment*

add()

add_model()

autocalled_initialize()

inject_wms_into()

prediction()

self_test()

set_experiment_environment()

class alpenglow.cpp.**RandomChoosingCombinedModelExpertUpdaterParameters**
Bases: sip.wrapper

eta

loss_type

top_k

class alpenglow.cpp.**RandomChoosingCombinedModelExpertUpdater**
Bases: *alpenglow.cpp.Updater*, *alpenglow.cpp.WMSUpdater*, *alpenglow.cpp.Initializable*, *alpenglow.cpp.NeedsExperimentEnvironment*

autocalled_initialize()

self_test()

set_experiment_environment()

set_wms()

update()

class alpenglow.cpp.**Evaluator**
Bases: sip.wrapper

get_loss()

get_score()

self_test()

class alpenglow.cpp.**CombinedModelParameters**
Bases: sip.wrapper

log_file_name

log_frequency

use_user_weights

class alpenglow.cpp.**CombinedModel**
Bases: *alpenglow.cpp.Model*

add()

add_model()

prediction()

class alpenglow.cpp.**RandomChoosingCombinedModel**
Bases: *alpenglow.cpp.Model*, *alpenglow.cpp.Initializable*, *alpenglow.cpp.NeedsExperimentEnvironment*

add()

add_model()

autocalled_initialize()

inject_wms_into()

prediction()

self_test()

set_experiment_environment()

class alpenglow.cpp.**ExternalModelParameters**
Bases: sip.wrapper

mode

class alpenglow.cpp.**ExternalModel**
Bases: *alpenglow.cpp.Model*

add()

clear()

prediction()

read_predictions()

self_test()

class alpenglow.cpp.**SimilarityModel**
Bases: sip.wrapper

self_test()

similarity()

class alpenglow.cpp.**ModelGradientUpdater**
Bases: sip.wrapper

beginning_of_updating_cycle()

end_of_updating_cycle()

self_test()

update()

class alpenglow.cpp.**ModelMultiUpdater**
Bases: sip.wrapper

self_test()

update()

class alpenglow.cpp.**Model**
Bases: sip.wrapper

add()

clear()

prediction()

read()

self_test()

write()

class alpenglow.cpp.**MassPredictor**
Bases: sip.wrapper

predict()

set_model()

7.13 implicit_data_creator

7.14 Filters

This is the filters header file.

class alpenglow.cpp.**AvailabilityFilter**
Bases: *alpenglow.cpp.ModelFilter*

This is the docstring for AvailabilityFilter. This filter filters the set of available items based on (time,itemId,duration) triplets. These have to be preloaded before

Sample code

```
def some_function():
  interesting = False
  print 'This line is highlighted.'
  print 'This one is not...'
  print '...but this one is.'
```

```
# this is python code
f = rs.AvailabilityFilter()
f.add_availability(10,1,10) #item 1 is available in the time interval (10,20)
```

active()

add_availability()

run(*rec_dat*)
Summary line.

Extended description of function.

Parameters

- **arg1** (*int*) – Description of arg1
- **arg2** (*str*) – Description of arg2

Returns Description of return value

Return type bool

self_test()

class alpenglow.cpp.**DummyModelFilter**
Bases: *alpenglow.cpp.ModelFilter*, *alpenglow.cpp.NeedsExperimentEnvironment*, *alpenglow.cpp.Initializable*

autocalled_initialize()

run()

self_test()

set_experiment_environment()

set_items()

set_users()

class alpenglow.cpp.**FactorModelFilter**
Bases: *alpenglow.cpp.ModelFilter*, *alpenglow.cpp.NeedsExperimentEnvironment*

autocalled_initialize()

get_global_items()

get_global_users()

run()

self_test()

set_experiment_environment()

set_items()

set_model()

set_users()

class alpenglow.cpp.**ModelFilter**
Bases: sip.wrapper

active()

run()

self_test()

7.15 ranking

class alpenglow.cpp.**RankComputerParameters**
Bases: sip.wrapper

random_seed

top_k

class alpenglow.cpp.**RankComputer**
Bases: *alpenglow.cpp.NeedsExperimentEnvironment*, *alpenglow.cpp.Initializable*

autocalled_initialize()

get_rank()

self_test()

set_experiment_environment()

set_model()

set_model_filter()

set_top_pop_container()

set_train_matrix()

7.16 offline_learners

class alpenglow.cpp.**OfflineEigenFactorModelALSLearnerParameters**
Bases: sip.wrapper

alpha

clear_before_fit

implicit

number_of_iterations

regularization_lambda

class alpenglow.cpp.**OfflineEigenFactorModelALSLearner**
Bases: *alpenglow.cpp.OfflineLearner*

fit()

iterate()

self_test()

set_copy_from_model()

set_copy_to_model()

set_model()

class alpenglow.cpp.**OfflineLearner**
Bases: sip.wrapper

fit()

self_test()

class alpenglow.cpp.**OfflineExternalModelLearnerParameters**
Bases: sip.wrapper

in_name_base

mode

out_name_base

class alpenglow.cpp.**OfflineExternalModelLearner**
Bases: *alpenglow.cpp.OfflineLearner*

fit()

set_model()

class alpenglow.cpp.**OfflineIteratingOnlineLearnerWrapperParameters**
Bases: sip.wrapper

number_of_iterations

seed

shuffle

class alpenglow.cpp.**OfflineIteratingOnlineLearnerWrapper**
Bases: *alpenglow.cpp.OfflineLearner*

add_early_updater()

add_iterate_updater()

add_updater()

fit()

self_test()

- genindex

- Module index
- search

Bibliography

[Hu2008] Hu, Yifan, Yehuda Koren, and Chris Volinsky. "Collaborative filtering for implicit feedback datasets." Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on. Ieee, 2008.

[Paterek2007] Arkadiusz Paterek. „Improving regularized singular value decomposition for collaborative filtering". In: Proc. KDD Cup Workshop at SIGKDD'07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining. San Jose, CA, USA, 2007, pp. 39–42.

[Koren2009] Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." Computer 42.8 (2009).

[X.He2016] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua. Fast matrix factorization for online recommendation with implicit feedback. In SIGIR, pages 549–558, 2016.

[Rendle2012] Rendle, Steffen. "Factorization machines with libfm." ACM Transactions on Intelligent Systems and Technology (TIST) 3.3 (2012): 57.

[Sarwar2001] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In Proc. WWW, pages 285–295, 2001.

[Ding2005] Y. Ding and X. Li. Time weight collaborative filtering. In Proc. CIKM, pages 485–492. ACM, 2005.

[Koren2008] Y. Koren, "Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model," Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, ACM Press, 2008, pp. 426-434.

Python Module Index

a

Index

A

B

C

D

G

H

I

L

M

N

O

P

R

S

T