Auto-Surprise

Release [0.1.2]

Rohan Anand, Joeran Beel

May 03, 2022

USAGE GUIDE

Auto-Surprise is an easy-to-use python AutoRecommenderSystem (AutoRecSys) library. It automates algorithm selection and hyperparameter tuning to build an optimized recommendation model. It uses the popular scikit library Surprise for recommender algorithms and Hyperopt for hyperparameter tuning.

Unfortunately, currently only linux systems are supported, but you can use WSL in windows as well.

To get started with Auto-Surprise, check out the *Quick Start* guide. If you have any issues or doubts, head over to the Github repository and create an issue.

CHAPTER

ONE

QUICK START

1.1 Installing

You will require Python >=3.6 and a linux based OS. With pip, installing Auto-Surprise is as easy as

```
pip install auto-surprise
```

Thats it. You are ready to get started

1.2 Quick Example

Here's a quick example of using Auto-Surprise to determine the best algorithm and hyperparameters for the Movielens 100k dataset.

```
# Import required libraries
from surprise import Dataset
from auto_surprise.engine import Engine

# Load the dataset
data = Dataset.load_builtin('ml-100k')

# Intitialize auto surprise engine
engine = Engine(verbose=True)

# Start the trainer
best_algo, best_params, best_score, tasks = engine.train(
    data=data,
    target_metric='test_rmse',
    cpu_time_limit=60 * 60, # Run for 1 hour
    max_evals=100
)
```

Thats it, after about 1 hour you should have the best algorithm along with the best parameters. To learn more, continue with the *Manual*

CHAPTER

TWO

MANUAL

Here, we will cover in more detail the usage for Auto-Surprise. We will start with an example, and go through each section

```
import hyperopt
from surprise import Reader, Dataset
from auto_surprise.engine import Engine

# Load the movielens dataset
file_path = os.path.expanduser('./ml-100k/u.data')
reader = Reader(line_format='user item rating timestamp', sep='\t', rating_scale=(1, 5))
data = Dataset.load_from_file(file_path, reader=reader)

# Intitialize auto surprise engine
engine = Engine(verbose=True)

# Start the trainer
best_algo, best_params, best_score, tasks = engine.train(
    data=data,
    target_metric='test_rmse',
    cpu_time_limit=60*60*2,
    max_evals=100,
    hpo_algo=hyperopt.tpe.suggest
)

# Build the model using the best algorithm and hyperparameters
best_model = engine.build_model(best_algo, best_params)
```

2.1 Loading the dataset

Auto-Surprise requires your dataset to be an instance of *surprise.dataset.DatasetAutoFolds*. You can learn more about this by reading the Surprise Dataset Docs

2.2 Initializing Auto-Surprise Engine

Engine is the main class for Auto-Surprise. You will need to initialize it once before you start using it.

```
engine = Engine(verbose=True, algorithms=['svd', 'svdpp', 'knn_basic', 'knn_baseline'])
```

- *verbose*: By default set to *True*. Controls the verbosity of Auto-Surprise.
- *algorithms*: The algorithms to be optimized. Must be in the form of an array of strings. Available choices are *['svd', 'svdpp', 'nmf', 'knn_basic', 'knn_baseline', 'knn_with_means', 'knn_with_z_score', 'co_clustering', 'slope_one', 'baseline_only']*
- *random_state*: Takes *numpy.random.RandomState*. Set this, as well as *random.seed* and *numpy.seed*, to make experiments reproducible.

2.3 Starting the Optimization process

To start the optimization method, you can use the *train* method of *Engine*. This will return the best algorithm, hyperparameters, best score, and tasks completed.

```
best_algo, best_params, best_score, tasks = engine.train(
    data=data,
    target_metric='test_rmse',
    cpu_time_limit=60*60*2,
    max_evals=100,
    hpo_algo=hyperopt.tpe.suggest
)
```

There are a few parameters you can use.

- *data*: The data as an instance of *surprise.dataset.DatasetAutoFolds*.
- *target_metric*: The metric we seek to minimize. Available options are *test_rmse* and *test_mae*.
- *cpu_time_limit*: The time limit we want to train. This is in seconds. For datasets like Movielens 100k, 1-2 hours is sufficient. But you may want to increase this based on the size of your dataset
- *max_evals*: The maximum number of evaluations each algorithm gets for hyper parameter optimization.
- *hpo_algo*: Auto-Surprise uses Hyperopt for hyperparameter tuning. By default, it's set to use TPE, but you can change this to any algorithm supported by hyperopt, such as Adaptive TPE or Random search.

2.4 Building the best Model

You can use the best alogithm and best hypermaters you got from the *train* method to build a model.

```
best_model = engine.build_model(best_algo, best_params)
```

You can pickle this model to save it and use it elsewhere.

CHAPTER

THREE

REPRODUCING EXPERIMENTS

You may want to make sure that your results are reproducible. This can be done easily by setting the seed and random state when initializing *Engine*.

```
from auto_surprise.engine import Engine

random.seed(123)
numpy.random.seed(123)

# Intitialize auto surprise engine with random state set
engine = Engine(verbose=True, random_state=numpy.random.RandomState(123))
```

This will make sure that you're results will be exactly the same, provided you're other training params also stay the same.

CHAPTER

FOUR

EVALUATION

We tested Auto-Surprise against 3 datasets

- Movielens 100k
- Jester Dataset 2 (100k Random sample)
- Book Crossing (100k random sample)

We then ran all surprise algorithms in their default configuration. We then ran Auto-Surprise with a time limit set to 2 hours and the target metric as RMSE. We also compared our results to gridsearch on a smaller search space.

CHAPTER

FIVE

RESULTS

Table 1: Results for Movielens 100k Dataset

Algorithm	RMSE	MAE	Time
Normal Predictor	1.5195	1.2200	00:00:01
SVD	0.9364	0.7385	00:00:23
SVD++	0.9196	0.7216	00:14:23
NMF	0.9651	0.7592	00:00:25
Slope One	0.9450	0.7425	00:00:15
KNN Basic	0.9791	0.7738	00:00:18
KNN with Means	0.9510	0.7490	00:00:19
KNN with Z-score	0.9517	0.7470	00:00:21
KNN Baseline	0.9299	0.7329	00:00:22
Co-clustering	0.9678	0.7581	00:00:08
Baseline Only	0.9433	0.7479	00:00:01
GridSearch	0.9139	0.7167	27:02:48
Auto-Surprise (TPE)	0.9136	0.7280	02:00:01
Auto-Surprise (ATPE)	0.9116	0.7244	02:00:02

Table 2: Results for Jester 2 Dataset (100k Random Sample)

Algorithm	RMSE	MAE	Time
Normal Predictor	7.277	5.886	00:00:01
SVD	4.905	3.97	00:00:13
SVD++	5.102	4.055	00:00:29
NMF	–	–	–
Slope One	5.189	3.945	00:00:02
KNN Basic	5.078	4.034	00:02:14
KNN with Means	5.124	3.955	00:02:16
KNN with Z-score	5.219	3.955	00:02:20
KNN Baseline	4.898	3.896	00:02:14
Co-clustering	5.153	3.917	00:00:12
Baseline Only	4.849	3.934	00:00:01
GridSearch	4.7409	3.8147	80:52:35
Auto-Surprise (TPE)	4.6489	3.6837	02:00:10
Auto-Surprise (ATPE)	4.6555	3.6906	02:00:01

Table 3: Results for Book Crossing Dataset (100k Random Sample)

Algorithm	RMSE	MAE	Time
Normal Predictor	4.8960	3.866	00:00:01
SVD	3.5586	3.013	00:00:11
SVD++	3.5842	2.991	00:01:48
NMF	–	–	–
Slope One	–	–	–
KNN Basic	3.9108	3.562	00:00:38
KNN with Means	3.8574	3.301	00:00:35
KNN with Z-score	3.8526	3.292	00:00:37
KNN Baseline	3.6181	3.101	00:00:36
Co-clustering	4.0168	3.409	00:00:19
Baseline Only	3.5760	3.095	00:00:02
GridSearch	3.5467	2.9554	48:29:46
Auto-Surprise (TPE)	3.5221	2.8871	02:00:58
Auto-Surprise (ATPE)	3.5190	2.8739	02:00:06

We see an improvement of anywhere from 0.8 - 4.0 % in RMSE using Auto-Surprise. The time taken to evaluate is also significantly less when compared to GridSearch.