Auto-Surprise

Release [0.1.2]

Rohan Anand, Joeran Beel

Jun 23, 2020

USAGE GUIDE:

1 Quick Start **3**
1.1 Installing 3
1.2 Quick Example 3

2 Manual **5**
2.1 Loading the dataset 6
2.2 Initializing Auto-Surprise Engine 6
2.3 Starting the Optimization process 6
2.4 Building the best Model 6

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

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(debug=False)

# 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(debug=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(debug=True)
```

Currently only the parameter available is *debug*, which will raise any caught exceptions instead of trying to continue the process

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, hyper-parameters, 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.