Surprise: A Python library for recommender systems

Nicolas Hug

1 Columbia University, Data Science Institute, New York City, New York, United States of America

Summary

Recommender systems aim at providing users with a list of recommendations of items that a service offers. For example, a video streaming service will typically rely on a recommender system to propose a personalized list of movies or series to each of its users. A typical problem in recommendation is that of rating prediction: given an incomplete dataset of user-item interactions which take the form of numerical ratings (e.g. on a scale from 1 to 5), the goal is to predict the missing ratings for all remaining user-item pairs.

Surprise is a Python library for building and analyzing rating prediction algorithms. It was designed to closely follow the scikit-learn API (Buitinck et al., 2013; Pedregosa et al., 2011), which should be familiar to users acquainted with the Python machine learning ecosystem.

Surprise provides a collection of estimators (or prediction algorithms) for rating prediction. Among others, classical algorithms are implemented such as the main similarity-based algorithms (Aggarwal & others, 2016), as well as algorithms based on matrix factorization like SVD (Koren, Bell, & Volinsky, 2009) or NMF (Lee & Seung, 2001). It also supports tools for model evaluation like cross-validation iterators and built-in metrics à la scikit-learn, as well as tools for model selection and automatic hyper-parameter search, namely grid search and randomized search. Thanks to simple primitives and a light API, users can also implement their own recommendation technique with a minimal amount of code.

Classical datasets such as the MovieLens datasets (Harper & Konstan, 2015) are directly available in the package, but user-defined datasets are also supported either by loading csv files, or by using pandas dataframes (McKinney, 2010).

Surprise is mainly written in Python, while the computationally intensive parts are optimized with Cython (Behnel et al., 2011). Internally, Surprise relies on built-in Python data structures (mainly dictionaries) as well as numpy arrays (Walt, Colbert, & Varoquaux, 2011).

Surprise was designed to be useful to researchers who want to quickly explore new recommendation ideas by supporting the creation of custom prediction algorithms, but can also serve as a learning resource for students and less experienced users thanks to its detailed documentation.

Other popular recommendation libraries with similar functionalities include LibRec (Guo, Zhang, Sun, & Yorke-Smith, n.d.) (Java) or MyMediaLite (Gantner, Rendle, Freudenthaler, & Schmidt-Thieme, 2011) (C#). In Python, OpenRec (Yang, Bagdasaryan, Gruenstein, Hsieh, & Estrin, 2018) and Spotlight (Kula, 2017) support neural-network inspired algorithms; implicit\(^1\) is specialized in implicit feedback recommendation, and LightFM (Kula, 2015) implements a hybrid algorithm based on matrix factorization. To the best of our knowledge, Surprise is the only library to provide a scikit-learn like API with model selection tools, and with a focus on explicit rating prediction.

\(^{1}\)https://github.com/benfred/implicit

Example

Here is a simple example showing how to (down)load a dataset, split it into five folds for cross-validation, and compute the Mean Average Error (MAE) and the Root Mean Squared Error (RMSE) of the SVD algorithm.

```python
from surprise import SVD
from surprise import Dataset
from surprise.model_selection import cross_validate

# Load the movielens-100k dataset (download it if needed).
data = Dataset.load_builtin('ml-100k')

# Use the famous SVD algorithm, with default parameters.
algo = SVD()

cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

# printed output:
# Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

# | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5 | Mean  | Std  |
# |--------|--------|--------|--------|--------|-------|------|
# | RMSE   | 0.9311 | 0.9370 | 0.9320 | 0.9317 | 0.9391 | 0.9342 |
# | MAE    | 0.7350 | 0.7375 | 0.7341 | 0.7342 | 0.7375 | 0.7357 |
# | Fit time | 6.53   | 7.11   | 7.23   | 7.15   | 3.99   | 6.40 |
# | Test time | 0.26   | 0.26   | 0.25   | 0.15   | 0.13   | 0.21 |

Acknowledgements

We are grateful to all the people who have contributed to the software, with special thanks to Maher Malaeb and David Stevens for the hyper-parameter searches, and to Lauriane Ducasse for the logo design.

References


