Tutorial: Open Source Online Learning Recommenders

Róbert Pálovics Domokos Kelen András A. Benczúr
Institute for Computer Science and Control
Hungarian Academy of Sciences (MTA SZTAKI)
{rpalovics,kelen.domokos.miklos,benczur}@sztaki.hu

ABSTRACT

Recommender systems have to serve in online environments that can be non-stationary. Traditional recommender algorithms may periodically rebuild their models, but they cannot adjust to quick changes in trends caused by timely information. In contrast, online learning models can adopt to temporal effects, hence they may overcome the effect of concept drift.

In our tutorial, we present open source systems capable of updating their models on the fly after each event: Apache Spark, Apache Flink and Alpenglow, a C++ based Python recommender framework. Participants of the tutorial will be able to experiment with all the three systems by using interactive Jupyter and Zeppelin Notebooks. Our final objective is to compare and then blend batch and online methods to build models providing high quality top-k recommendation in non-stationary environments.

KEYWORDS

open source recommender systems; temporal evaluation; ranking prediction by online learning; streaming; concept drift

TARGET AUDIENCE

Recommender systems researchers and practitioners. Basic Python and/or Scala programming skills are required.

EXPERIENCE OF THE PRESENTERS

The presenters have been using Jupyter for international courses on data science relying on tools such as Scikit-learn, Spark, Flink and formerly GraphLab Create. They participated in the organization of the RecSys Challenges 2016 and 2017. In addtion, they contribute to the Apache Flink Machine Learning Library.

TUTORIAL URL

https://github.com/rpalovics/recsys-2017-online-learning-tutorial

SUPPORT

From the *EU H2020* research and innovation program under grant agreement *Streamline No 688191* and the "Big Data—Momentum" grant of the Hungarian Academy of Sciences.

RecSys '17, , August 27-31, 2017, Como, Italy

ACM ISBN 978-1-4503-4652-8/17/08.
https://doi.org/http://dx.doi.org/10.1145/3109859.3109937

INTRODUCTION

In a real recommender system, users request one or a few items at a time, and may get exposed to new information that can change their needs and taste before they return to the service the next time. Therefore, *top item recommendation by online learning* is usually more relevant than the Netflix Prize style batch rating prediction [3, 7]. In this tutorial, we consider top-k recommendation in highly non-stationary environments. Our goal is to promptly update recommender models after user interactions by online learning methods, and investigate whether performance gains in batch experiments carry over to online environments. In addition, we consider data sets with *implicit feedback* [3].

We set up environments, in which events and requests for recommendation arrive in a continuous data stream. The difficulty of evaluating *streaming recommenders* was first mentioned in [5], although they split the stream into a training and a testing part instead of evaluating continuously online. Ideas for *online evaluation metrics* appeared first in [6, 7, 11]. An important part of the tutorial considers the framework for evaluating recommender systems over streaming data. We rely on ideas of [1, 7] for the online DCG measure.

The objectives of the tutorial are to

- provide evaluation measures suited to non-stationary environments that may also indicate which algorithm would work well on a particular data set,
- present open source online learning recommender systems in Jupyter and Zeppelin notebooks,
- compare the performance of simpler algorithms that can be updated online—and hence use the most recent data as well—with more complex algorithms that can be updated only periodically,
- experiment with combinations of batch and online training to improve accuracy.

TEMPORAL EVALUATION

In a time-aware or online recommender that potentially re-trains its model after each new item, we have to generate a new top-k recommendation list for every single event in the evaluation period [1]. In an online setting, as seen in Figure 1, whenever we see a new user-item pair, we assume that the user becomes active and requests a recommendation. We recommend items of potential interest for the user, which then we match against the actual item consumed. After this evaluation step, we may update the model. Following [2], we propose DCG instead of precision and recall for measuring the quality of top-k recommendation. Furthermore, in our time-aware ranking evaluation setting DCG is computed individually for each event and then averaged in time[6, 7].

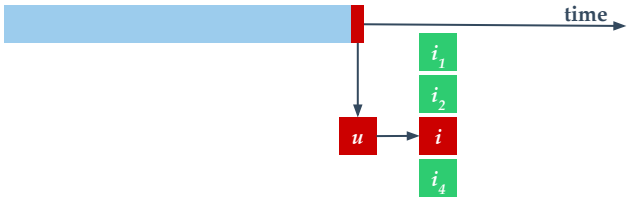

Figure 1: Temporal evaluation of the online ranking prediction problem.

OPEN SOURCE RECOMMENDERS

The tutorial covers recommender algorithms both by batch and by online learning. Participants have the opportunity to install and evaluate the following systems and algorithms in Jupyter and Zeppelin notebooks:

Alpenglow[1] is a free and open source C++ based framework with easy-to-use Python API especially suited for conjoint batch and online learning. We experiment in Aplenglow with
- non-personalized temporal popularity and item-to-item recommender models,
- time-aware variants of nearest neighbor [9],
- SGD based batch and online matrix factorization [10],
- asymmetric matrix factorization [8].

Flink[2] matrix factorization algorithms are available only as pull requests. We experiment both with batch and online factorization models,
- batch iALS[3] [10],
- batch DSGD[4] [4],
- parameter server based asynchronous online SGD[5].

The first two methods use the Flink batch API while the third uses the Flink streaming API.

SparkML[6] provides batch iALS[7] that we compare to the batch models of Alpenglow and Flink.

TUTORIAL OUTLINE

All material (slides, readings, Jupyter and Zeppelin notebooks, code) used in the tutorial will be publicly available after the conference. The instructors aid the participants with the installation. Basic knowledge of Python and Scala is assumed. The material includes
- slides explaining the online ranking prediction problem, the temporal evaluation framework, and the models,
- installation guides for Alpenglow, Flink and SparkML,
- experiments in Jupyter and Zeppelin notebooks,
- public research data collections.

The tutorial consists of two independent sessions, with the first one focusing on the main concepts and Alpenglow, and the the second one focusing on Apache Spark and Flink recommenders. In the first part, participants may
- install Alpenglow,
- experiment with various batch and online recommendation models and evaluate them in a temporal setting,
- compare and then combine batch and online matrix factorization methods.

In the second session, we provide distributed recommendation models that may learn from potentially large data sets *and* update their models by online learning. The audience will
- install Flink and Spark, and compare their batch iALS implementations and the batch DSGD included in the Flink Batch API,
- compare the performance of the above batch models to the Parameter Server based asynchronous online SGD in Flink,
- combine batch and online matrix factorization using the Flink Streaming API.

[1] https://github.com/rpalovics/Alpenglow
[2] https://flink.apache.org/
[3] https://github.com/apache/flink/pull/2542
[4] https://github.com/apache/flink/pull/2819
[5] https://github.com/gaborhermann/\flink-parameter-server
[6] https://spark.apache.org/docs/latest/ml-guide.html
[7] https://spark.apache.org/docs/latest/ml-guide.html

REFERENCES

[1] Jacob Abernethy, Kevin Canini, John Langford, and Alex Simma. 2007. Online collaborative filtering. *University of California at Berkeley, Tech. Rep* (2007).

[2] Azzah Al-Maskari, Mark Sanderson, and Paul Clough. 2007. The relationship between IR effectiveness measures and user satisfaction. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 773–774.

[3] Xavier Amatriain and Justin Basilico. 2016. Past, Present, and Future of Recommender Systems: An Industry Perspective. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 211–214.

[4] Rainer Gemulla, Erik Nijkamp, Peter J Haas, and Yannis Sismanis. 2011. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 69–77.

[5] Neal Lathia, Stephen Hailes, and Licia Capra. 2009. Temporal collaborative filtering with adaptive neighbourhoods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 796–797.

[6] Róbert Pálovics and András A Benczúr. 2013. Temporal influence over the Last.fm social network. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. ACM, 486–493.

[7] Róbert Pálovics, András A Benczúr, Levente Kocsis, Tamás Kiss, and Erzsébet Frigó. 2014. Exploiting temporal influence in online recommendation. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 273–280.

[8] Arkadiusz Paterek. 2007. 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*. 39–42. http://serv1.ist.psu.edu:8080/viewdoc/summary; jsessionid=CBC0A80E61E800DE518520F9469B2FD1?doi=10.1.1.96.7652

[9] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*. ACM Press, New York, NY, USA, 285–295. https://doi.org/10.1145/371920.372071

[10] G. Takács, I. Pilászy, B. Németh, and D. Tikk. 2008. Investigation of various matrix factorization methods for large recommender systems. In *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*. ACM, 1–8.

[11] João Vinagre, Alípio Mário Jorge, and João Gama. 2014. Evaluation of recommender systems in streaming environments. In *Workshop on Recommender Systems Evaluation: Dimensions and Design (REDD 2014), held in conjunction with RecSys 2014. October 10, 2014, Silicon Valley, United States.*