Journal of Machine Learning Research 18 (2018) 1-5 Submitted 7/17; Revised 4/18; Published 4/18

`tick`: a Python Library for Statistical Learning, with an emphasis on Hawkes Processes and Time-Dependent Models

Emmanuel Bacry EMMANUEL.BACRY@POLYTECHNIQUE.EDU
Martin Bompaire MARTIN.BOMPAIRE@POLYTECHNIQUE.EDU
Philip Deegan PHILIP.DEEGAN@POLYTECHNIQUE.EDU
Stéphane Gaïffas STEPHANE.GAIFFAS@POLYTECHNIQUE.EDU
Søren V. Poulsen SOREN.POULSEN@POLYTECHNIQUE.EDU
Centre de Mathématiques Appliquées
École polytechnique
UMR 7641, 91128 Palaiseau, France

Editor: Geoff Holmes

Abstract

This paper introduces `tick`, is a statistical learning library for Python 3, with a particular emphasis on time-dependent models, such as point processes, tools for generalized linear models and survival analysis. The core of the library provides model computational classes, solvers and proximal operators for regularization. It relies on a C++ implementation and state-of-the-art optimization algorithms to provide very fast computations in a single node multi-core setting. Source code and documentation can be downloaded from `https://github.com/X-DataInitiative/tick`.

Keywords: Statistical Learning; Python; Hawkes processes; Optimization; Generalized linear models; Point Process; Survival Analysis

1. Introduction

The aim of the `tick` library is to provide for the Python community a large set of tools for statistical learning, previously not available in any framework. Though `tick` focuses on time-dependent modeling, it actually introduces a set of tools that go way beyond this particular set of models, thanks to a highly modular optimization toolbox. It benefits from thorough documentation (including tutorials with many examples), and a strongly tested Python API that brings to the scientific community cutting-edge algorithms with a high level of customization. Optimization algorithms such as SVRG (Johnson and Zhang, 2013) or SDCA (Shalev-Shwartz and Zhang, 2013) are among the several optimization algorithms available in `tick` that can be applied (in a modular way) to a large variety of models. An emphasis is placed on time-dependent models: from the Cox regression model (Andersen et al., 2012), a very popular model in survival analysis, to Hawkes processes, used in a wide range of applications such as geophysics (Ogata, 1988), finance (Bacry et al., 2015) and more recently social networks (Zhou et al., 2013; Xu et al., 2016). To the best of our knowledge, `tick` is the most comprehensive library that deals with Hawkes processes, since it brings parametric and nonparametric estimators of these models to a new accessibility level.

2. Existing Libraries

The `tick` library follows, whenever possible, `scikit-learn`'s API (Pedregosa et al., 2011; Buitinck et al., 2013) which is well-known for its completeness and ease of use. However, while `scikit-learn` targets a wide spectrum, `tick` has a more specific objective: implementing highly-optimized algorithms with a particular emphasis on time-dependent modeling (not proposed in `scikit-learn`). The `tick` optimization toolbox relies on state-of-the-art optimization algorithms, and is implemented in a very modular way. It allows more possibilities than other `scikit-learn` API based optimization libraries such as `lightning`[1].

A wide variety of time-dependent models are taken care of by `tick`, which makes it the most comprehensive library that deals with Hawkes processes for instance, by including the main inference algorithms from the literature. Despite the growing interest in Hawkes models, very few open source packages are available. There are mainly three of them. The library `pyhawkes`[2] proposes a small set of Bayesian inference algorithms for Hawkes process. `hawkes` R[3] is an R-based library that provides a single estimation algorithm, and is hardly optimized. Finally, `PtPack`[4] a C++ library which proposes parametric maximum likelihood estimators, with sparsity-inducing regularizations. However, since `tick` is a Python library, it addresses a different community to PtPack. Moreover, as illustrated below, `tick` provides better performance than `PtPack`.

3. Package Architecture

The `tick` library has four main modules: `tick.hawkes` for Hawkes processes (see Section 4 for a detailed review), `tick.linear_model` with linear, logistic and Poisson regression, `tick.robust` for robust linear models and `tick.survival` for survival analysis. Each of these modules provide simulation tools and learners to easily learn from data. The core of `tick` is made of easy to combine penalization techniques (`tick.prox` module) and several convex solvers (`tick.solver`), to train almost any available model in the library, see Table 1 for a non-exhaustive list of possible combinations. An exhaustive list is available on the documentation web page[5], and is given in Figure 7 of the supplementary material.

4. Hawkes

Distributing a comprehensive open source library for Hawkes processes is one of the primary aims of the `tick` library: it provides many non-parametric and parametric estimation algorithms that are listed in Table 2 as well as simulation tools for many kernel types as shown in Figure 5 of the supplementary material. This diversity of algorithms is illustrated in Figure 1 in which we show how two kernels of different shapes are estimated by four different algorithms. A first use case for modeling high-frequency financial data is given in Figure 2 (with the associated Python code), while a second use-case about propagation analysis of earthquake aftershocks can be found in Figure 3.

1. `http://contrib.scikit-learn.org/lightning`
2. `https://github.com/slinderman/pyhawkes`
3. `https://cran.r-project.org/web/packages/hawkes/hawkes.pdf`
4. `https://github.com/dunan/MultiVariatePointProcess`
5. `https://x-datainitiative.github.io/tick/`

Model	Proximal operator	Solver
Linear regression	SLOPE	Gradient Descent
Logistic regression	L1 (Lasso)	Stochastic Variance Reduced Gradient
Poisson regression	Total Variation	Stochastic Gradient Descent
Cox regression	Group L1	Accelerated Gradient Descent
Hawkes with exp. kernels	L2 (Ridge)	Stochastic Dual Coordinate Ascent

Table 1: `tick` allows the user to combine many models, prox and solvers

Non Parametric	Parametric
EM (Lewis and Mohler, 2011)	Single exponential kernel
Basis kernels (Zhou et al., 2013)	Sum of exponentials kernels
Wiener-Hopf (Bacry and Muzy, 2014)	Sum of gaussians kernels (Xu et al., 2016)
NPHC (Achab et al., 2017)	ADM4 (Zhou et al., 2013)

Table 2: Models and estimation techniques for Hawkes processes available in `tick`

5. Benchmarks

In Figure 4, we benchmark computational times for both simulation and estimation of Hawkes processes (with exponential kernels) using `hawkes` R (where only simulation is available), `PtPack` and `tick` on respectively 2, 4 and 16 cores. The model fits in plots "Fit" and "Multicore fit" are compared on simulated 16-dimensional Hawkes processes, with an increasing number of events: small=5×10^4, medium=2×10^5, large=10^6, xlarge= 5×10^7, while 200, 400 and 750 dimensional Hawkes processes are fitted in plot "High-dimensional fitting". We observe that `tick` outperforms by several orders of magnitudes both `hawkes` R and `PtPack`, in particular for large datasets. Benchmarks against `scikit-learn` for logistic regression are also provided in Figure 6 from the supplementary material.

Acknowledgments

The authors thank the Data-science Initiative of École polytechnique and Intel® for supporting `tick` development

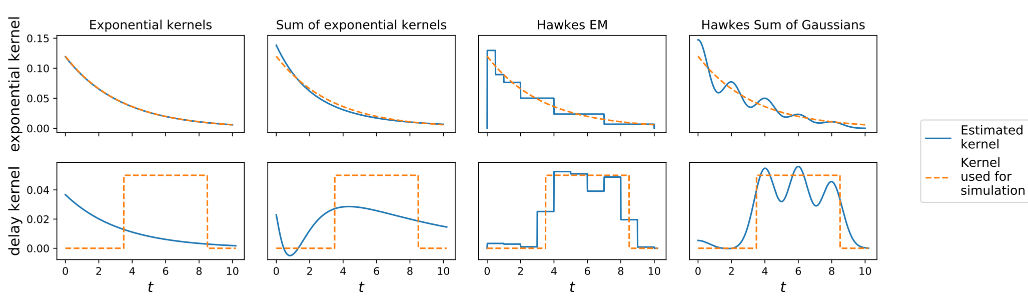

Figure 1: Illustration of different kernel shapes and estimations obtained by `tick` on two 1-dimensional simulated Hawkes processes.

```python
timestamps_list = fetch_hawkes_bund_data()

hawkes_learner = HawkesConditionalLaw(
  claw_method="log", quad_method="log", n_quad=50,
  delta_lag=0.1, min_lag=5e-4, max_lag=500,
  min_support=1e-4, max_support=1, n_threads=4)

hawkes_learner.fit(timestamps_list)

plot_hawkes_kernel_norms(
  hawkes_learner, node_names=["P_u", "P_d", "T_a", "T_b"])
```

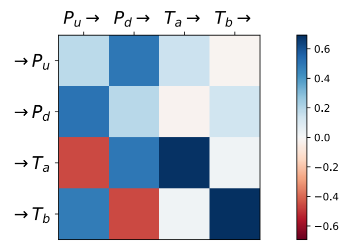

Figure 2: Kernel norms of a Hawkes process fitted on high-frequency financial data from the Bund market (Bacry et al., 2016) where P_u (resp. P_d) counts the number of upward (resp. downward) mid-price moves and T_a (resp. T_b) counts the number of market orders at the ask (resp. bid) that do not move the price.

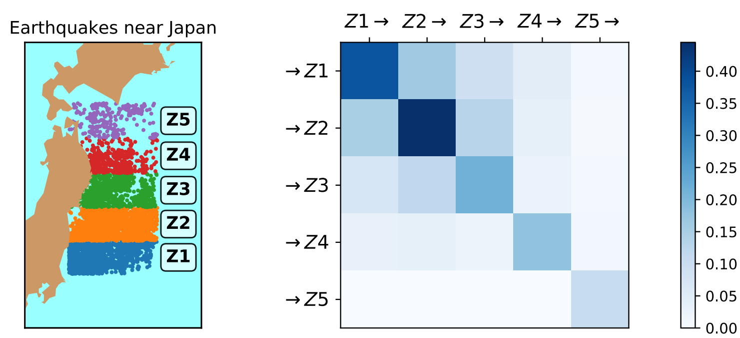

Figure 3: Modeling of earthquake propagation with Hawkes processes on a dataset from Ogata (1988). The left hand side gives the location of the earthquakes while right hand side illustrates the propagation matrix, namely how likely an earthquake in a given zone will trigger an aftershock in another zone.

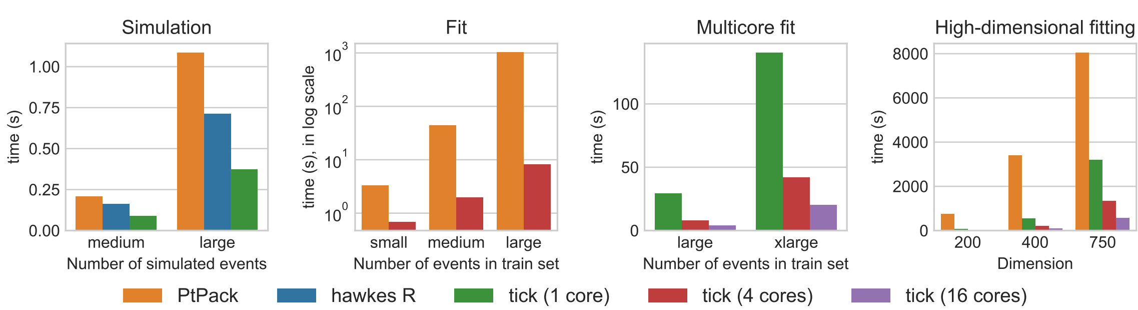

Figure 4: Computational timings of `tick` versus `PtPack` and `hawkes` R. `tick` strongly outperforms both libraries for simulation and fitting (note that the "Fit" graph is in log-scale). "Multicore fit" and "High-dimensional fitting" plots show that `tick` benefits from multi-core environments to speed up computations.

References

M. Achab, E. Bacry, S. Gaïffas, I. Mastromatteo, and J.-F. Muzy. Uncovering causality from multivariate hawkes integrated cumulants. In *International Conference on Machine Learning*, pages 1–10, 2017.

P. K. Andersen, O. Borgan, R. D. Gill, and N. Keiding. *Statistical Models based on Counting Processes*. Springer Science, 2012.

E. Bacry and J.-F. Muzy. Second order statistics characterization of hawkes processes and non-parametric estimation. *arXiv preprint arXiv:1401.0903*, 2014.

E. Bacry, I. Mastromatteo, and J.-F. Muzy. Hawkes processes in finance. *Market Microstructure and Liquidity*, 1(01):1550005, 2015.

E. Bacry, T. Jaisson, and J.-F. Muzy. Estimation of slowly decreasing hawkes kernels: application to high-frequency order book dynamics. *Quantitative Finance*, 16(8):1179–1201, 2016.

L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J VanderPlas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

E. Lewis and G. Mohler. A nonparametric em algorithm for multiscale hawkes processes. *preprint*, pages 1–16, 2011.

Y. Ogata. Statistical models for earthquake occurrences and residual analysis for point processes. *Journal of the American Statistical Association*, 83(401):9–27, 1988.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013.

H. Xu, M. Farajtabar, and H. Zha. Learning granger causality for hawkes processes. In *Proceedings of International Conference on Machine Learning*, pages 1717–1726, 2016.

K. Zhou, H. Zha, and L. Song. Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes. In *AISTATS*, volume 31, pages 641–649, 2013.