tick: a Python library for statistical learning, with a particular emphasis on time-dependent modeling

Emmanuel Bacry emmanuel.bacry@polytechnique.edu
Martin Bompaire martin.bompaire@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: xxxx

Abstract

tick is a statistical learning library for Python 3, with a particular emphasis on time-dependent models, such as point processes, and tools for generalized linear models and survival analysis. The core of the library is an optimization module providing model computational classes, solvers and proximal operators for regularization. tick 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 propose to 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 allow to go way beyond this particular set of models, thanks to a highly modular optimization toolbox. It benefits from a thorough documentation (including tutorials with many examples), and a strongly tested 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 done 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 theses models to a new accessibility level.

2. Existing libraries

tick follows, whenever possible, the scikit-learn API (Pedregosa et al., 2011; Buitinck et al., 2013) which is well-known for its completeness and ease of use, which makes it the reference Python machine learning library. 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 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 a R-based library that provides a single estimation algorithm, and is hardly optimized. Finally, PtPack[4] is a C++ library which proposes mainly parametric maximum likelihood estimators, with sparsity-inducing regularizations. However, PtPack is not interfaced with a user-friendly scripting language such as Python, which makes it less accessible to end-users for quick prototyping and experimenting on datasets. Moreover, as illustrated below, PtPack exhibits poor performance compared to tick.

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 regression and tick.survival for survival analysis. Each of these modules provide simulation tools and learners to easily learn from data. Whenever possible, tick follows the scikit-learn API. The core of tick is made of easy to combine penalization techniques (proximal operators), available in the tick.prox module and several convex solvers, available in the 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 6 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 as well as simulation tools for many kernel types, that are listed in Table 2. This diversity of algorithms is illustrated in Figure 1 (with the associated Python code) in which we show how two kernels of different shapes are estimated by four different algorithms. A

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	L2 (Ridge)	Gradient Descent
Logistic regression	L1 (Lasso)	Accelerated Gradient Descent
Poisson regression	Total Variation	Stochastic Gradient Descent
Cox regression	Group L1	Stochastic Variance Reduced Gradient
Hawkes with exp. kernels	SLOPE	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: Hawkes estimation algorithms implemented in `tick`

first use case for modeling high-frequency financial data is given in Figure 2, while a second use-case about propagation analysis of earthquake aftershocks can be found in Figure 4.

5. Benchmarks

We perform benchmark tests for both simulation and estimation of Hawkes processes (with exponential kernels) using `tick`, `hawkes` R (where only simulation is available) and `PtPack`, on respectively 2, 4 and 16 cores. In Figure 3 we compare computational times for simulation and fitting of Hawkes processes. The model fits 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. We observe on this experiment 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 5 from the supplementary material.

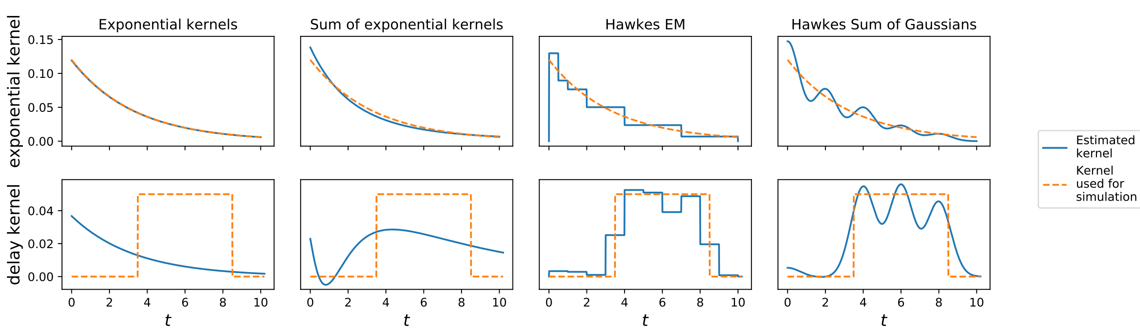

Figure 1: Illustration of different kernels shapes and estimations obtained by `tick` on two 1D simulated Hawkes processes with intensity kernels displayed with dashed orange lines.

```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"])
```

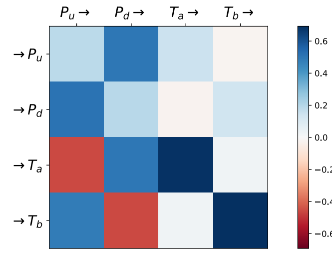

Figure 2: Kernels norms of a Hawkes process fitted on high-frequency financial data from the Bund market (Bacry et al., 2016) where P_u (resp. P_d) counts the number of upward (resp. downward) mid-price moves and T_a (resp. T_b) counts the number of market orders at the ask (resp. bid) that do not move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

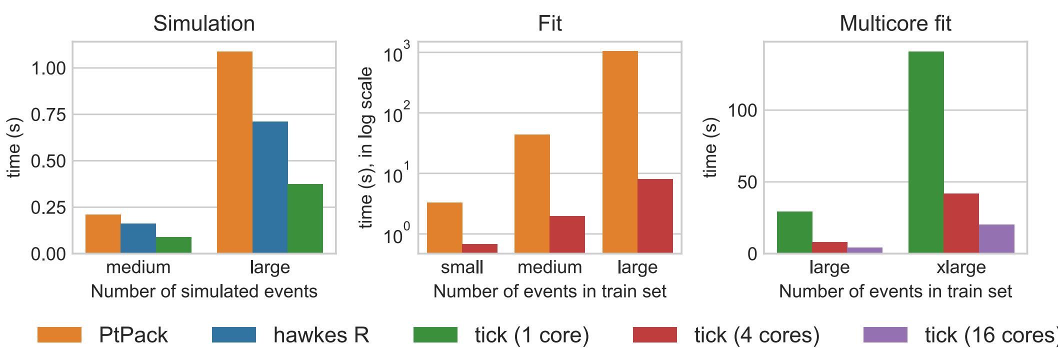

Figure 3: Computational timings of `tick` versus `PtPack` and `hawkes` R. `tick` strongly outperforms both libraries for simulation and fitting (note that fit graph is in log-scale). Third figure shows that `tick` benefits from multi-core environments to speed up computations.

Acknowledgments

We would like to acknowledge support for this project from the Datascience Initiative of École polytechnique and Intel® for supporting `tick` development.

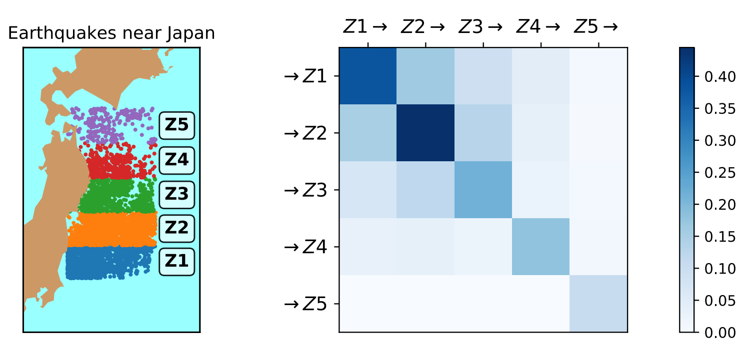

Figure 4: Analysis with Hawkes processes of earthquake propagation with a dataset from Ogata (1988). On the left we can see where earthquakes have occurred and on the right the propagation matrix, i.e. how likely a earthquake in a given zone will trigger an aftershock in another zone. We can observe than zone 1, 2 and 3 are tightly linked while zone 4 and 5 are more self-excited.

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 triggering kernels for multi-dimensional hawkes processes. In *Proceedings of the International Conference on Machine Learning*, pages 1301–1309, 2013.

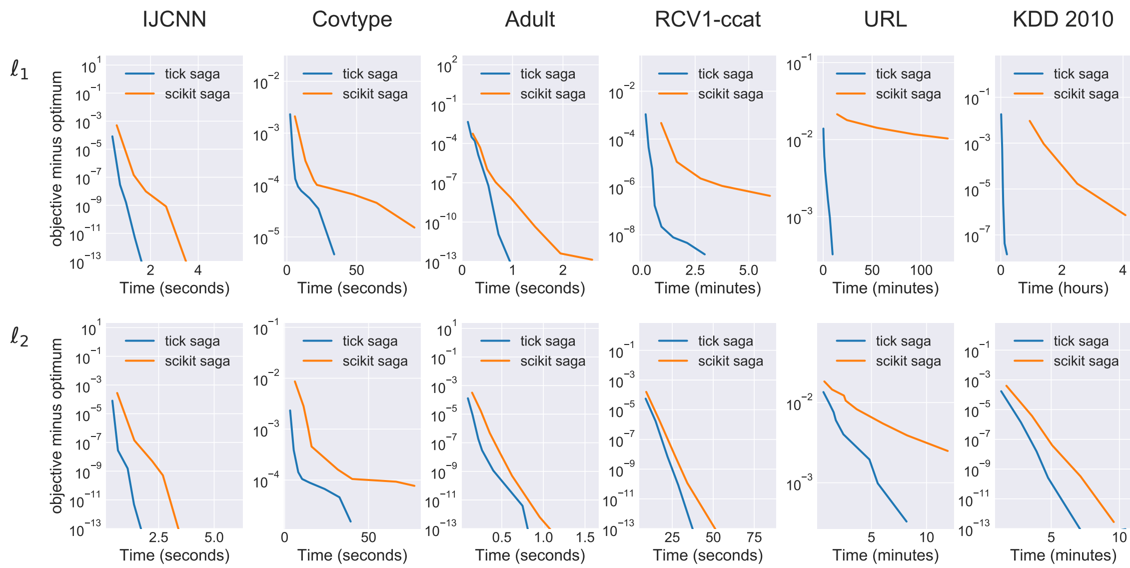

Figure 5: Speed comparison with `scikit-learn` library. These plots display time needed to achieve a given precision for logistic regression with ℓ_1 and ℓ_2 penalizations on commonly used datasets. In both cases we use SAGA solver as the two libraries provide it.

Appendix

A. Speed comparison

We compare fitting results for binary logistic regression with `tick` and `scikit-learn`. These experiments are run on commonly used datasets described in Table 3. Note that Covtype has been standardized, hence the first two datasets IJCNN and Covtype are dense and the last four datasets are sparse. Two types of penalization have been tested: ℓ_1 (Lasso) and ℓ_2 (Ridge). In both cases the regularization parameter λ has been set to $1/n$ where n is the number of samples and we have left the default step-size for both libraries. Results are given in Figure 5. Overall, `tick` is slightly faster because it makes faster iterations: both libraries reach the same objective after each pass over the data but `tick` performs these computations faster. Also, ℓ_1 penalization in high dimension is difficult for `scikit-learn` (see URL and KDD 2010) whereas `tick` handles it without any additional problem.

dataset	# samples	# features	density
IJCNN	141,691	22	100 %
Covtype	581,012	54	100 %
Adult	32,561	123	11.3 %
RCV1-ccat	804,414	47,236	0.0016 %
URL	2,396,130	3,231,961	0.000036 %
KDD 2010	19,264,097	1,163,024	0.00078 %

Table 3: Datasets used to perform binary logistic regression.

B. Package structure

The package structure is detailed in Figure 6. We retrieve all the following modules:

- `tick.hawkes` : Inference and simulation of Hawkes processes, with both parametric and non-parametric estimation techniques and flexible tools for simulation. It is split in three submodules: `tick.hawkes.inference`, `tick.hawkes.simulation`, `tick.hawkes.model`.
- `tick.linear_model` : Inference and simulation of linear models, including among others linear, logistic and Poisson regression, with a large set of penalization techniques and solvers.
- `tick.robust` : Tools for robust inference. It features tools for outliers detection and models such as Huber regression, among others robust losses.
- `tick.survival` : Inference and simulation for survival analysis, including Cox regression with several penalizations.
- `tick.prox` : Proximal operators for penalization of models weights. Such an operator can be used with (almost) any model and any solver.
- `tick.solver` : A module that provides a bunch of state-of-the-art optimization algorithms, including both batch and stochastic solvers
- `tick.dataset` : Provides easy access to datasets used as benchmarks in tick.
- `tick.plot` : Some plotting utilities used in tick, such as plots for point processes and solver convergence.

Figure 6: Structure of `tick` package