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

Emmanuel Bacry  
Martin Bompaire  
Stéphane Gaïffas  
Søren V. Poulsen
Centre de Mathématiques Appliquées  
École polytechnique  
UMR 7641, 91128 Palaiseau, France

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 these models to a new accessibility level.
2. Existing libraries

tick follows, when 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 many more possibilities than many other scikit-learn API based optimization libraries such as lightning.

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. Despite the growing interest in Hawkes models, very few open source packages are available. There are mainly three of them. The Python-based library pyhawkes only implements Hawkes algorithms included in published works of their contributors. hawkes R is a R-based library that provides a single estimation algorithm, and is hardly optimized. Finally, PtPack includes both classic and more exotic estimation algorithms (most of them are provided by tick), however its C++ implementation makes it far less easy to use than tick, and exhibits poor performance compared to tick, as illustrated below.

3. Package architecture

The tick library is organized in several modules which interact with each other. Each module aims to solve a specific problem. The main module is the tick.inference module that performs easy model fitting and the tick.optim module that allows decoupling the way models are fitted by specifying independently statistical model information, regularization and solver. There are many other smaller modules in tick. The tick.simulation module provides efficient simulations of many models, the tick.plot module allows graphical representations of many tick objects, the tick.dataset module allows very simple access to common ready-to-use real datasets and the tick.preprocessing module allows many ways of preprocessing of the data for easier model fitting.

3.1 Inference module

Inference classes are split in three main categories: linear models with linear, logistic and Poisson regression, survival analysis with Cox regression and finally Hawkes processes inference (see Section 4 for a detailed review). Following scikit-learn API, inference is performed by calling the fit method (see Example 1) in all cases.

2. https://github.com/slinderman/pyhawkes
4. https://github.com/dunan/MultiVariatePointProcess
5. These datasets are hosted on the https://github.com/X-DataInitiative/tick-datasets repository
6. A deeper customization of the methods, such as combination of proximal operators, or fine-tuning of several stochastic solvers, is accessible using the optimization module tick.optim.
from tick.dataset import fetch_hawkes_bund_data
from tick.inference import HawkesConditionalLaw
from tick.plot import plot_hawkes_kernel_norms

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 1: Kernels norms of a Hawkes process fitted on high-frequency financial data from
the Bund market. This reproduces experiments run in (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.

3.2 Optimization module

This optimization module allows to combine models (tick.optim.model), penalizations
(tick.optim.prox) and solvers (tick.optim.solver) in many ways. A non-exhaustive
list of classes are provided in table 1.

<table>
<thead>
<tr>
<th>Model</th>
<th>Proximal operator</th>
<th>Solver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear regression</td>
<td>L2 (Ridge)</td>
<td>Gradient Descent</td>
</tr>
<tr>
<td>Logistic regression</td>
<td>L1 (Lasso)</td>
<td>Accelerated Gradient Descent</td>
</tr>
<tr>
<td>Poisson regression</td>
<td>Total Variation</td>
<td>Stochastic Gradient Descent</td>
</tr>
<tr>
<td>Cox regression</td>
<td>Nuclear</td>
<td>Stochastic Variance Reduced Gradient</td>
</tr>
<tr>
<td>Hawkes with exp. kernels</td>
<td>SLOPE</td>
<td>Stochastic Dual Coordinate Ascent</td>
</tr>
</tbody>
</table>

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

4. Hawkes

Distributing an open source library including a lot of material 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.

<table>
<thead>
<tr>
<th>Non Parametric</th>
<th>Parametric</th>
</tr>
</thead>
<tbody>
<tr>
<td>EM (Lewis and Mohler, 2011)</td>
<td>Single exponential kernel</td>
</tr>
<tr>
<td>Basis kernels (Zhou et al., 2013)</td>
<td>Sum of exponentials kernels</td>
</tr>
<tr>
<td>Wiener-Hopf (Bacry and Muzy, 2014)</td>
<td>Sum of gaussians kernels (Xu et al., 2016)</td>
</tr>
<tr>
<td></td>
<td>ADM4 (Zhou et al., 2013)</td>
</tr>
</tbody>
</table>

Table 2: Hawkes estimation algorithms implemented in tick
Figure 2: Illustration of different kernels shapes and estimations obtained by tick.

Figure 3: Benchmarks for Hawkes processes with exponential kernels. tick strongly outperforms other libraries, and benefits from multi-core environments.

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. Results are displayed in Figure 3 where time is shown versus total number of events of the Hawkes process. Our conclusion is that tick outperforms by several orders of magnitudes hawkes R and PtPack, in particular for large datasets.

Acknowledgments

We would like to acknowledge support for this project from the Datascience Initiative of École polytechnique.
References


