tick: a Python library for statistical learning, with an emphasis on Hawkes processes and time-dependent modeling

—

Supplementary material

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: xxxx

A. Simulation of Hawkes processes with tick

A Hawkes process (Hawkes and Oakes, 1974) is a multivariate point-process. Namely, it models timestamps, also called ticks $\{t_k^i\}_{i\geq 1}$ of nodes $i = 1, \dots, D$ using a multivariate counting process $N_t = [N_t^1 \cdots N_t^D]$, for $t \geq 0$, where $N_t^i = \sum_{k\geq 1} \mathbf{1}_{t_k^i \leq t}$ for any $t \geq 0$. In a Hawkes process the intensity of component N^i has the following auto-regressive structure:

$$\lambda_i(t) = \mu_i + \sum_{j=1}^{D} \int \phi_{ij}(t-s) dN_j(s) = \mu_i + \sum_{j=1}^{D} \sum_{k : t_k^j < t} \phi_{ij}(t - t_k^j).$$

The $\mu_i \geq 0$ are called *baselines* intensities, and correspond to the exogenous intensity of events from node i, and ϕ_{ij} for $1 \leq i, j \leq D$ are called *kernels*, that quantify the influence of past events from node j on the intensity of events from node i. The main parametric model for the kernels is the so-called *exponential* kernel, in which we consider $\phi_{ij}(t) = \alpha_{ij}\beta \exp(-\beta t)$ for $\alpha_{ij} > 0$ and $\beta > 0$. In this model $A = [\alpha_{i,j}]_{1\leq i,j\leq d}$ is understood as an *adjacency matrix*, since entry $A_{i,j} \geq 0$ quantifies the impact of the activity of node j on the activity of node i, while $\beta > 0$ is a memory parameter.

The tick library provides simulation tools for several types of Hawkes processes, given the kernel choice: exponential, sum of exponential, power-law, or the linear-approximation of any kernel function. This variety of simulation possibilities is illustrated in Figure 5.

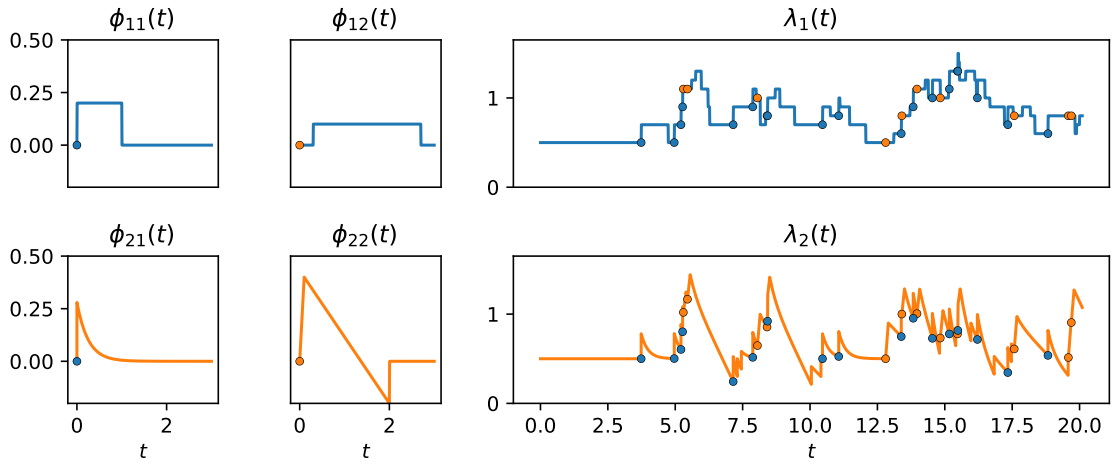

Figure 5: Simulation of a 2-dimensional Hawkes process. The 4 kernels are shown on the left hand side. The simulated intensities are illustrated on the right hand side (against time, up to time 20), where events (ticks) are represented by colored dots (blue corresponding to node 1 and orange to node 2).

B. Computational timings: `tick` vs `scikit-learn` to train a logistic regression

We compare computational timings to fit a logistic regression with `tick` and `scikit-learn`. These experiments are run on commonly used datasets (including large scale ones) 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), since these are the only ones proposed by `scikit-learn`. In both cases the regularization parameter λ has been set to $1/n$ where n is the number of samples, and we consider the default step-sizes proposed by both libraries. For a fair comparison, we considered the SAGA algorithm, see Defazio et al. (2014), to minimize the penalized logistic regression objective, since both libraries implement it. Results are given in Figure 6. Overall, `tick` is much faster since it makes faster iterations: both libraries reach roughly the same objective after each pass over the data (since both of them use the same algorithm, only their implementations differ). Moreover, ℓ_1 penalization in high dimension is difficult for `scikit-learn` (see URL and KDD 2010) whereas `tick` handles it without any additional problem.

C. `tick` structure

The structure of the `tick` library is detailed in Figure 7. Here is a short description of the contents of each module:

- `tick.hawkes` : Inference and simulation of Hawkes processes, with both parametric and non-parametric estimation techniques and flexible tools for simulation. It

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.

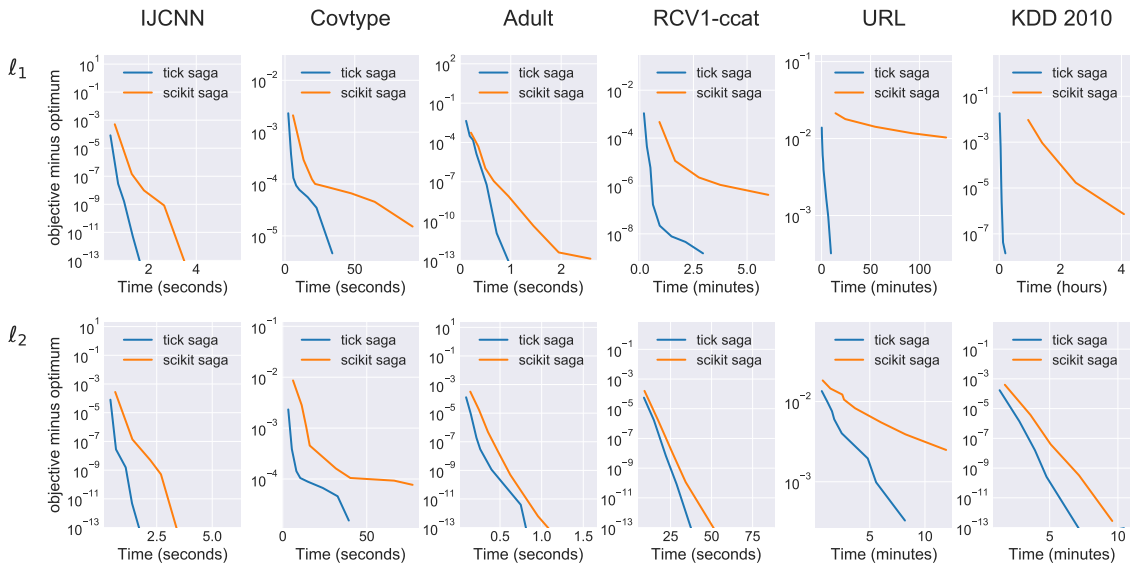

Figure 6: Speed comparison with `scikit-learn` library. These plots display time needed to achieve a given precision for logistic regression with ℓ_1 and ℓ_2 penalizations on commonly used datasets. In both cases we use SAGA solver as the two libraries provide it.

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.

References

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pages 1646–1654, 2014.

Alan G Hawkes and David Oakes. A cluster process representation of a self-exciting process. *Journal of Applied Probability*, 11(3):493–503, 1974.

Figure 7: Structure of the `tick` library