




HiddenMarkovModels.jl: generic, fast and reliable state space modeling

Guillaume Dalle ^{1,2,3}

¹ Information, Learning and Physics laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL), Station 11, CH-1015 Lausanne ² Information and Network Dynamics laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL), Station 14, CH-1015 Lausanne ³ Statistical Physics of Computation laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne

DOI: [10.21105/joss.06436](https://doi.org/10.21105/joss.06436)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Mehmet Hakan Satman](#) 

Reviewers:

- [@DanielRivasMD](#)
- [@dmbates](#)

Submitted: 12 September 2023

Published: 05 April 2024

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

Hidden Markov Models (or HMMs) are a very popular statistical framework, with numerous applications ranging from speech recognition to bioinformatics. They characterize a sequence of *observations* Y_1, \dots, Y_T by assuming the existence of a hidden sequence of *states* X_1, \dots, X_T . The distribution of a state X_t can only depend on the previous state X_{t-1} , and the distribution of an observation Y_t can only depend on the current state X_t . In addition, both of these dynamics may be influenced by exogenous control variables U_1, \dots, U_T . This is a very versatile and practical set of assumptions: see Rabiner (1989) for an introduction, Cappé et al. (2005) for a book-length treatment and Bengio & Frasconi (1994) for a seminal discussion of HMMs with controls.

Given a sequence of observations and a parametric family of HMMs \mathbb{P}_θ , there are several problems one can face. In generic graphical models, these problems are often intractable, but HMMs have a tree-like structure that yields exact solution procedures with polynomial complexity. The package `HiddenMarkovModels.jl` leverages the Julia language (Bezanson et al., 2017) to implement those algorithms in a *generic, fast and reliable* way.

Inference problem	Algorithm
Best state sequence $\arg\max_{X_{1:T}} \mathbb{P}_\theta(X_{1:T} Y_{1:T}, U_{1:T})$	Viterbi
Observation sequence likelihood $\mathbb{P}_\theta(Y_{1:T} U_{1:T})$	Forward
State marginals $\mathbb{P}_\theta(X_t Y_{1:T}, U_{1:T})$	Forward-backward
Maximum likelihood parameter $\arg\max_\theta \mathbb{P}_\theta(Y_{1:T} U_{1:T})$	Baum-Welch

Statement of need

The initial motivation for `HiddenMarkovModels.jl` was an application of HMMs to reliability analysis for the French railway company SNCF (Dalle, 2022). In this industrial use case, the observations were marked temporal point processes (sequences of timed events with structured metadata) generated by condition monitoring systems, possibly influenced by the daily activity of the train unit.

Unfortunately, nearly all implementations of HMMs we surveyed (in Julia and Python) expect the observations to be generated by a *predefined set of distributions*, with *no temporal heterogeneity*. In Julia, the previous reference package `HMMBase.jl` (Mouchet, 2023) requires compliance with the `Distributions.jl` (Besançon et al., 2021) interface, which precludes

anything not scalar- or array-valued, let alone point processes. In Python, the numpy-based `hmmlearn` ([hmmlearn developers, 2023](#)) and the PyTorch-based `pomegranate` ([Schreiber, 2018, 2014/2024](#)) each offer a catalogue of discrete and continuous distributions, but do not allow for easy extension by the user. The more recent JAX-based `dynamax` ([Chang et al., 2022/2024; Murphy, 2023; Särkkä & Svensson, 2023](#)) is the only package adopting an extensible interface with optional controls, similar to ours.

Focusing on Julia specifically, other downsides of `HMMBase.jl` include the lack of support for *multiple observation sequences*, *automatic differentiation*, *sparse transition matrices* or *number types beyond 64-bit floating point*. Two other Julia packages each provide a subset of functionalities that `HMMBase.jl` lacks, namely `HMMGradients.jl` ([Antonello, 2021](#)) and `MarkovModels.jl` ([Ondel et al., 2022](#)), but they are less developed and ill-suited to uninformed users.

Package design

`HiddenMarkovModels.jl` was designed to overcome the limitations mentioned above, following a few guiding principles.

Our package is *generic*. Observations can be arbitrary objects, and the associated distributions only need to implement two methods: a loglikelihood `logdensityof(dist, x)` and a sampler `rand(rng, x)`. Number types are not restricted, and automatic differentiation of the sequence loglikelihood ([Qin et al., 2000](#)) is supported both in forward and reverse mode, partly thanks to `ChainRulesCore.jl` ([White et al., 2022](#)). The extendable `AbstractHMM` interface allows incorporating features such as priors or structured transitions, as well as temporal or control dependency, simply by redefining three methods:

```
initialization(hmm)
transition_matrix(hmm, control)
obs_distributions(hmm, control)
```

Our package is *fast*. Julia's blend of multiple dispatch and just-in-time compilation delivers satisfactory speed even when working with unexpected types that Python's tensor backends could not easily handle. Inference routines rely on BLAS calls for linear algebra, and exploit multithreading to process sequences in parallel.

Our package is *reliable*. It is thoroughly tested and documented, with an extensive API reference and accessible tutorials. Special care was given to code quality, type stability, and compatibility checks with various downstream packages (like automatic differentiation packages).

However, our package is also *limited in scope*. It aims at CPU efficiency for moderately-sized state spaces, and remains untested on GPU. Furthermore, it does not manipulate probabilities in the logarithmic domain, but instead uses the scaling trick ([Rabiner, 1989](#)) with a variation borrowed from `HMMBase.jl`. Thus, its numerical stability might be worse than that of Python counterparts on challenging instances. Luckily, thanks to unrestricted number types, users are free to bring in third-party packages like `LogarithmicNumbers.jl` ([Rowley, 2023](#)) to recover additional precision.

Benchmarks

We compare `HiddenMarkovModels.jl`, `HMMBase.jl`, `hmmlearn`, `pomegranate` and `dynamax` on a test case with univariate Gaussian observations. The reason for this low-dimensional choice is to spend most of the time in the generic HMM routines themselves, as opposed to the loglikelihood computations which are problem-specific. The data consists of 50 independent sequences of length 100 each, with a number of states varying from 2 to 10, to which we apply all inference algorithms (with Baum-Welch performing 5 iterations).

All benchmarks were run in Julia version 1.10.2 with BenchmarkTools.jl (Chen & Revels, 2016), calling Python with PythonCall.jl (Rowley, 2022), and plotting results with CairoMakie.jl (Danisch & Krumbiegel, 2021). The comparison code imports HiddenMarkovModels.jl version 0.5.0 (commit f7cf63b), and it is accessible in the [libs/HMMComparison/](#) subfolder of our GitHub repository. We tried to minimize parallelism effects by running everything on a single thread, and made the assumption that the Julia-to-Python overhead is negligible compared to the algorithm runtime.

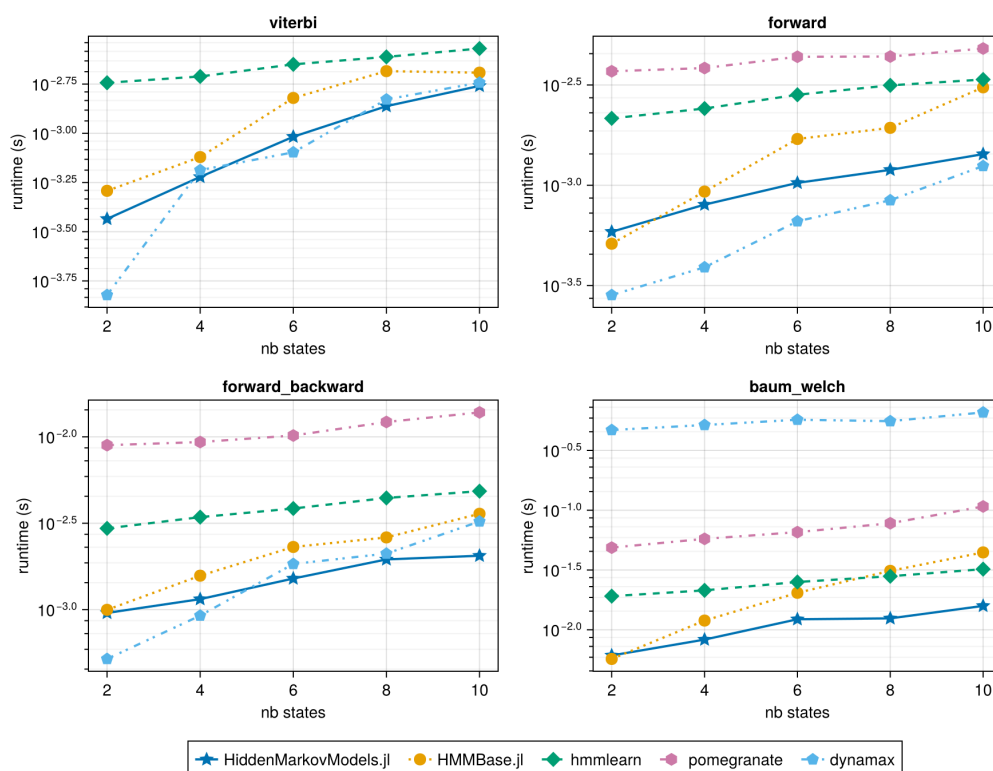


Figure 1: Benchmark of HMM packages

As we can see, HiddenMarkovModels.jl is the fastest option in Julia, and the second-fastest overall behind dynamax (we think the large runtimes of dynamax in Baum-Welch might stem from [incorrect benchmarks](#)). The key observation is that we achieved this speedup over HMMBase.jl while *simultaneously increasing generality* in half a dozen different ways.

Conclusion

HiddenMarkovModels.jl fills a longstanding gap in the Julia package ecosystem, by providing an efficient and flexible framework for state space modeling.

Acknowledgements

Work on this package started during my PhD at École des Ponts, in partnership with SNCF Réseau and SNCF Voyageurs, whose support I acknowledge. It continued during my postdoctoral position at EPFL.

My gratitude goes to Maxime Mouchet and Jacob Schreiber, the developers of HMMBase.jl and

pomegranate respectively, for their help and advice. In particular, Maxime agreed to designate `HiddenMarkovModels.jl` as the official successor to `HMMBase.jl`, for which I thank him.

References

- Antonello, N. (2021). *HMMGradients.jl: Enables computing the gradient of the parameters of Hidden Markov Models (HMMs)*. Zenodo. <https://doi.org/10.5281/zenodo.4454565>
- Bengio, Y., & Frasconi, P. (1994). An Input Output HMM Architecture. *Advances in Neural Information Processing Systems*, 7. <https://proceedings.neurips.cc/paper/1994/hash/8065d07da4a77621450aa84fee5656d9-Abstract.html>
- Besançon, M., Papamarkou, T., Anthoff, D., Arslan, A., Byrne, S., Lin, D., & Pearson, J. (2021). Distributions.jl: Definition and Modeling of Probability Distributions in the JuliaStats Ecosystem. *Journal of Statistical Software*, 98, 1–30. <https://doi.org/10.18637/jss.v098.i16>
- Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59(1), 65–98. <https://doi.org/10.1137/141000671>
- Cappé, O., Moulines, E., & Rydén, T. (2005). *Inference in Hidden Markov Models*. Springer New York. <https://doi.org/10.1007/0-387-28982-8>
- Chang, P., Harper-Donnelly, G., Kara, A., Li, X., Linderman, S., & Murphy, K. (2024). *Dynamax: State Space Models library in JAX*. Probabilistic machine learning. <https://github.com/probml/dynamax> (Original work published 2022)
- Chen, J., & Revels, J. (2016, August 15). *Robust benchmarking in noisy environments*. <https://doi.org/10.48550/arXiv.1608.04295>
- Dalle, G. (2022). *Machine learning and combinatorial optimization algorithms, with applications to railway planning* [PhD thesis, École des Ponts ParisTech]. <https://pastel.hal.science/tel-04053322>
- Danisch, S., & Krumbiegel, J. (2021). Makie.jl: Flexible high-performance data visualization for Julia. *Journal of Open Source Software*, 6(65), 3349. <https://doi.org/10.21105/joss.03349>
- hmmlearn developers. (2023). *Hmmlearn: Hidden Markov Models in Python, with scikit-learn like API*. hmmlearn. <https://github.com/hmmlearn/hmmlearn>
- Mouchet, M. (2023). *HMMBase.jl: Hidden Markov Models for Julia*. <https://github.com/maxmouchet/HMMBase.jl>
- Murphy, K. P. (2023). *Probabilistic Machine Learning: Advanced Topics*. The MIT Press. ISBN: 978-0-262-04843-9
- Ondel, L., Lam-Yee-Mui, L.-M., Kocour, M., Corro, C. F., & Burget, L. (2022). GPU-Accelerated Forward-Backward Algorithm with Application to Lattice-Free MMI. *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8417–8421. <https://doi.org/10.1109/ICASSP43922.2022.9746824>
- Qin, F., Auerbach, A., & Sachs, F. (2000). A Direct Optimization Approach to Hidden Markov Modeling for Single Channel Kinetics. *Biophysical Journal*, 79(4), 1915–1927. [https://doi.org/10.1016/S0006-3495\(00\)76441-1](https://doi.org/10.1016/S0006-3495(00)76441-1)
- Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 257–286. <https://doi.org/cswph2>
- Rowley, C. (2022). *PythonCall.jl: Python and Julia in harmony*. JuliaPy. <https://github.com/JuliaPy/PythonCall.jl>
- Rowley, C. (2023). *LogarithmicNumbers.jl: A logarithmic number system for Julia*. <https://github.com/cjdoris/LogarithmicNumbers.jl>

- Särkkä, S., & Svensson, L. (2023). *Bayesian Filtering and Smoothing* (2nd ed.). Cambridge University Press. <https://doi.org/10.1017/9781108917407>
- Schreiber, J. (2018). Pomegranate: Fast and Flexible Probabilistic Modeling in Python. *Journal of Machine Learning Research*, 18(164), 1–6. <http://jmlr.org/papers/v18/17-636.html>
- Schreiber, J. (2024). *Jmschrei/pomegranate*. <https://github.com/jmschrei/pomegranate> (Original work published 2014)
- White, L., Abbott, M., Zgubic, M., Revels, J., Arslan, A., Axen, S., Schaub, S., Robinson, N., Ma, Y., Dhingra, G., willtebbutt, Heim, N., Widmann, D., Rosemberg, A. D. W., Schmitz, N., Rackauckas, C., Heintzmann, R., frankschae, Fischer, K., ... Chorney, F. (2022). *JuliaDiff/ChainRules.jl: V1.23.0* (Version v1.23.0). Zenodo. <https://doi.org/10.5281/zenodo.5881966>