




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

3 Guillaume Dalle  1,2,3

4 1 Information, Learning and Physics laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL),
5 Station 11, CH-1015 Lausanne 2 Information and Network Dynamics laboratory, Ecole Polytechnique
6 Fédérale de Lausanne (EPFL), Station 14, CH-1015 Lausanne 3 Statistical Physics of Computation
7 laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne

DOI: 10.xxxxxx/draft

Software

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

Editor: Mehmet Hakan Satman 

Reviewers:

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

Submitted: 12 September 2023

Published: unpublished

License

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

8 Summary

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

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

Inference problem	Algorithm
Best state sequence $\operatorname{argmax}_{X_{1:T}} \mathbb{P}_\theta(X_{1:T} Y_{1:T}, U_{1:T})$	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 $\operatorname{argmax}_\theta \mathbb{P}_\theta(Y_{1:T} U_{1:T})$	Baum-Welch

23 Statement of need

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

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

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

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

44 Package design

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

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

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

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

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

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

68 Benchmarks

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

75 All benchmarks were run in Julia version 1.10.2 with BenchmarkTools.jl (Chen & Revels, 2016),
76 calling Python with PythonCall.jl (Rowley, 2022), and plotting results with CairoMakie.jl
77 (Danisch & Krumbiegel, 2021). The comparison code imports HiddenMarkovModels.jl version
78 0.5.0 (commit f7cf63b), and it is accessible in the [libs/HMMComparison/](#) subfolder of our
79 GitHub repository. We tried to minimize parallelism effects by running everything on a single
80 thread, and made the assumption that the Julia-to-Python overhead is negligible compared to
81 the algorithm runtime.

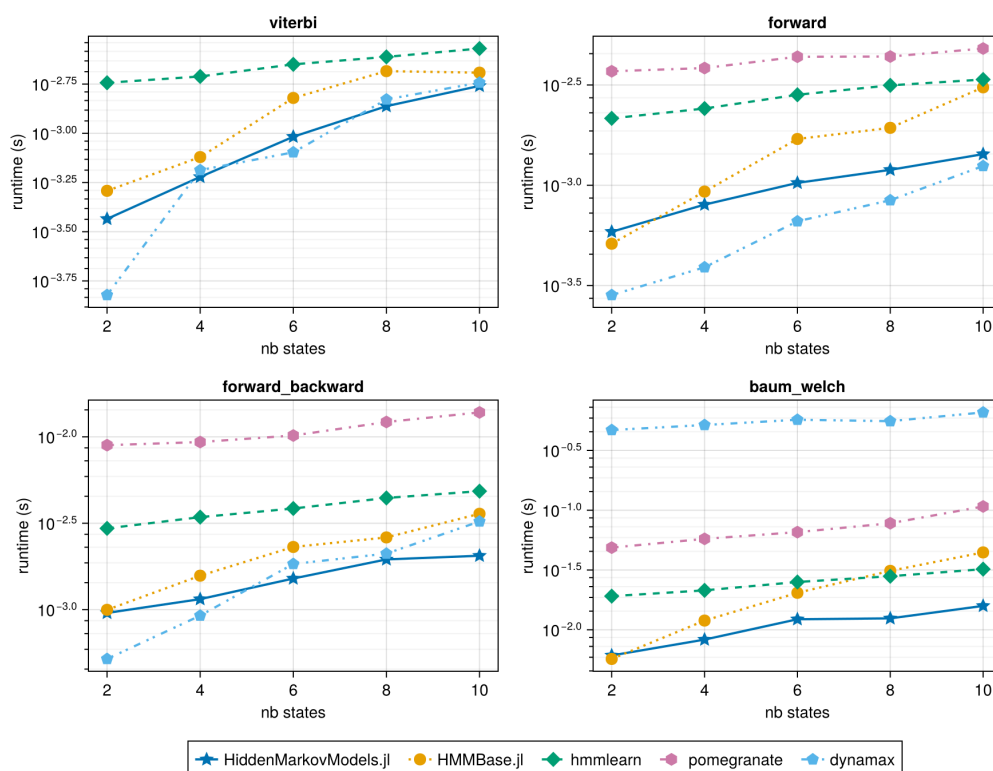


Figure 1: Benchmark of HMM packages

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

Conclusion

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

Acknowledgements

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

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

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

96 References

- 97 Antonello, N. (2021). *HMMGradients.jl: Enables computing the gradient of the parameters of*
98 *Hidden Markov Models (HMMs)*. Zenodo. <https://doi.org/10.5281/zenodo.4454565>
- 99 Bengio, Y., & Frasconi, P. (1994). An Input Output HMM Architecture. *Advances in Neural*
100 *Information Processing Systems*, 7. [https://proceedings.neurips.cc/paper/1994/hash/](https://proceedings.neurips.cc/paper/1994/hash/8065d07da4a77621450aa84fee5656d9-Abstract.html)
101 [8065d07da4a77621450aa84fee5656d9-Abstract.html](https://proceedings.neurips.cc/paper/1994/hash/8065d07da4a77621450aa84fee5656d9-Abstract.html)
- 102 Besançon, M., Papamarkou, T., Anthoff, D., Arslan, A., Byrne, S., Lin, D., & Pearson, J. (2021).
103 *Distributions.jl: Definition and Modeling of Probability Distributions in the JuliaStats*
104 *Ecosystem*. *Journal of Statistical Software*, 98, 1–30. <https://doi.org/10.18637/jss.v098.i16>
- 105 Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A Fresh Approach to
106 Numerical Computing. *SIAM Review*, 59(1), 65–98. <https://doi.org/10.1137/141000671>
- 107 Cappé, O., Moulines, E., & Rydén, T. (2005). *Inference in Hidden Markov Models*. Springer
108 New York. <https://doi.org/10.1007/0-387-28982-8>
- 109 Chang, P., Harper-Donnelly, G., Kara, A., Li, X., Linderman, S., & Murphy, K. (2024).
110 *Dynamax: State Space Models library in JAX*. Probabilistic machine learning. [https://](https://github.com/probml/dynamax)
111 github.com/probml/dynamax
- 112 Chen, J., & Revels, J. (2016). *Robust benchmarking in noisy environments* (No.
113 arXiv:1608.04295). arXiv. <https://doi.org/10.48550/arXiv.1608.04295>
- 114 Dalle, G. (2022). *Machine learning and combinatorial optimization algorithms, with applications*
115 *to railway planning* [PhD thesis, École des Ponts ParisTech]. [https://www.theses.fr/](https://www.theses.fr/2022ENPC0047)
116 [2022ENPC0047](https://www.theses.fr/2022ENPC0047)
- 117 Danisch, S., & Krumbiegel, J. (2021). Makie.jl: Flexible high-performance data visualization for
118 Julia. *Journal of Open Source Software*, 6(65), 3349. <https://doi.org/10.21105/joss.03349>
- 119 hmmlearn developers. (2023). *Hmmlearn: Hidden Markov Models in Python, with scikit-learn*
120 *like API*. hmmlearn. <https://github.com/hmmlearn/hmmlearn>
- 121 Mouchet, M. (2023). *HMMBase.jl: Hidden Markov Models for Julia*. [https://github.com/](https://github.com/maxmouchet/HMMBase.jl)
122 [maxmouchet/HMMBase.jl](https://github.com/maxmouchet/HMMBase.jl)
- 123 Ondel, L., Lam-Yee-Mui, L.-M., Kocour, M., Corro, C. F., & Burget, L. (2022). GPU-
124 Accelerated Forward-Backward Algorithm with Application to Lattice-Free MMI. *ICASSP*
125 *2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing*
126 *(ICASSP)*, 8417–8421. <https://doi.org/10.1109/ICASSP43922.2022.9746824>
- 127 Qin, F., Auerbach, A., & Sachs, F. (2000). A Direct Optimization Approach to Hidden
128 Markov Modeling for Single Channel Kinetics. *Biophysical Journal*, 79(4), 1915–1927.
129 [https://doi.org/10.1016/S0006-3495\(00\)76441-1](https://doi.org/10.1016/S0006-3495(00)76441-1)
- 130 Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech
131 recognition. *Proceedings of the IEEE*, 77(2), 257–286. <https://doi.org/cswph2>
- 132 Rowley, C. (2022). *PythonCall.jl: Python and Julia in harmony*. JuliaPy. [https://github.com/](https://github.com/JuliaPy/PythonCall.jl)
133 [JuliaPy/PythonCall.jl](https://github.com/JuliaPy/PythonCall.jl)
- 134 Rowley, C. (2023). *LogarithmicNumbers.jl: A logarithmic number system for Julia*. [https://](https://github.com/cjdoris/LogarithmicNumbers.jl)
135 github.com/cjdoris/LogarithmicNumbers.jl
- 136 Schreiber, J. (2018). Pomegranate: Fast and Flexible Probabilistic Modeling in Python. *Journal*
137 *of Machine Learning Research*, 18(164), 1–6. <http://jmlr.org/papers/v18/17-636.html>

138 White, F. C., Abbott, M., Zgubic, M., Revels, J., Axen, S., Arslan, A., Schaub, S., Robinson,
139 N., Ma, Y., Dhingra, G., Tebbutt, W., Heim, N., Widmann, D., Rosemberg, A. D. W.,
140 Schmitz, N., Rackauckas, C., Heintzmann, R., Frankschae, Noack, A., ... Vertechi, P. (2022).
141 *JuliaDiff/ChainRules.jl: V1.44.7*. Zenodo. <https://doi.org/10.5281/ZENODO.4754896>

DRAFT