JUCHMME: A Java Utility for Class Hidden Markov Models and Extensions for biological sequence analysis

Tamposis, Ioannis A.; Tsirigos, Konstantinos D.; Theodoropoulou, Margarita C.; Kontou, Panagiota I.; Tsaousis, Georgios N.; Sarantopoulou, Dimitra; Litou, Zoi I.; Bagos, Pantelis G.

Published in:
Bioinformatics

Link to article, DOI:
10.1093/bioinformatics/btz533

Publication date:
2020

Document Version
Publisher's PDF, also known as Version of record

Bioinformatics, 35(24), 2019, 5309–5312
doi: 10.1093/bioinformatics/btz533
Advance Access Publication Date: 28 June 2019
Applications Note

Sequence analysis

JUCHMME: a Java Utility for Class Hidden Markov Models and Extensions for biological sequence analysis

Ioannis A. Tamposis[1], Konstantinos D. Tsirigos [2], Margarita C. Theodoropoulou[1], Panagiota I. Kontou[1], Georgios N. Tsaousis[3], Dimitra Sarantopoulou [4], Zoi I. Litou[5] and Pantelis G. Bagos [1,*]

[1]Department of Computer Science and Biomedical Informatics, University of Thessaly, Lamia 35100, Greece, [2]Department of Health Technology, Section for Bioinformatics, Technical University of Denmark, Kgs Lyngby, Denmark, [3]Department of Bioinformatics, Genekor Medical SA, Athens, Greece, [4]Institute of Translational Medicine and Therapeutics, University of Pennsylvania, Philadelphia, PA, USA and [5]Section of Cell Biology and Biophysics, Department of Biology, School of Science, National and Kapodistrian University of Athens, Athens, Greece

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Received on March 4, 2019; revised on May 4, 2019; editorial decision on June 21, 2019; accepted on June 25, 2019

Abstract

Summary: JUCHMME is an open-source software package designed to fit arbitrary custom Hidden Markov Models (HMMs) with a discrete alphabet of symbols. We incorporate a large collection of standard algorithms for HMMs as well as a number of extensions and evaluate the software on various biological problems. Importantly, the JUCHMME toolkit includes several additional features that allow for easy building and evaluation of custom HMMs, which could be a useful resource for the research community.

Availability and implementation: http://www.compgen.org/tools/juchmme, https://github.com/pbagos/juchmme.

Contact: pbagos@compgen.org

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Hidden Markov Models (HMMs) are probabilistic models widely used in biological sequence analysis. The most commonly used type of HMM is the profile HMM (pHMM), which is especially useful for multiple sequence alignment and remote homology detection. Tools based on pHMMs like HMMER (Eddy, 1998) and SAM (Hughey and Krogh, 1996) are used routinely by thousands of researchers around the world. Software tools based on pHMM use a standard architecture in order to construct the models corresponding to the multiple alignment. However, for other applications in biological sequence analysis, the need of HMMs with custom architecture arises. For these applications, the model design can be time-consuming, thus software tools that allow easy implementation of HMMs with arbitrary and user-defined topologies are necessary. We present here an open-source tool, JUCHMME, which can be used to easily build custom HMMs utilizing the most complete collection of algorithms.

2 Materials and methods

JUCHMME can be used to build a broad range of HMMs. Standard HMMs are commonly trained by the Maximum Likelihood (ML) criterion using the Baum-Welch algorithm (Durbin *et al.*, 1998; Rabiner, 1989). Maximizing $P(\mathbf{x}|\theta)$ corresponds to *unsupervised learning*. JUCHMME implements also the gradient-descent algorithm proposed by Baldi and Chauvin (1994) and the Segmental *k*-means

algorithm also known as Viterbi learning algorithm (Juang and Rabiner, 1990). In most biological applications, the use of labeled sequences for training is advisable. In these models, each amino acid sequence **x** is accompanied by a sequence of labels **y**. Krogh named this model Class HMM (CHMM) and proposed simple modifications of forward and backward algorithms (Krogh, 1994). The likelihood to be maximized in such situations (*supervised learning*) could be the joint probability of the sequences and the labels given the model, $P(\mathbf{x}, \mathbf{y}|\theta)$, which corresponds to ML training, or the probability of the labels given the sequences, $P(\mathbf{y}|\mathbf{x},\theta)$, which corresponds to conditional maximum likelihood (CML) estimation that leads to discriminative training (Krogh, 1997). The gradient-based algorithms necessary for these models are implemented using various heuristics for fast and robust convergence (Bagos *et al.*, 2004a, b), including adaptive step-size and resilient back-propagation (RPROP). Moreover, a newly developed variant of the segmental *k*-means for labeled sequences, for both ML and CML, is also available (Theodoropoulou *et al.*, 2017).

The toolkit implements also a large collection of decoding algorithms such as the standard Viterbi, Posterior-Viterbi (Fariselli *et al.*, 2005), N-Best (Krogh, 1997) and Optimal Accuracy Posterior Decoder (Käll *et al.*, 2005). Moreover, modifications of the decoding algorithms that allow constrained predictions (i.e. fixing the observed labels) are also available (Bagos *et al.*, 2006). These can be helpful when one wishes to incorporate some prior knowledge, such as experimentally derived information, in the prediction process.

Additionally, to overcome HMM limitations, a number of extensions have been developed such as Hidden Neural Networks (HNNs) (Krogh and Riis, 1999), models that condition on previous observations (Tamposis *et al.*, 2018) and a newly developed method for semi-supervised learning of HMMs that can incorporate labeled, unlabeled and partially labeled data (Tamposis *et al.*, 2019).

Finally, the toolkit includes numerous auxiliary programs that automate several routine processes, like cross-validation tests (including jackknife), generating random sequences from a given model, various options for initializing the models (both HMMs and HNNs) suitable for testing purposes, and various programs for measuring prediction accuracy (Baldi *et al.*, 2000; Zemla *et al.*, 1999) and reliability (Melen *et al.*, 2003). Of note, JUCHMME also supports multicore parallelization a feature that can speed up the computations.

3 Usage scenario

To illustrate the utility and features of JUCHMME toolkit, we provide a simple real-life scenario (see Supplementary Material). For this we need to follow the steps:

1. define the transition probabilities, using a spreadsheet (Supplementary Fig. S2). This is important because non-zero elements define the allowed transitions and thus the model architecture.
2. define the initial emission probabilities, using a spreadsheet (Supplementary Fig. S3).
3. define the model file using a text editor (Supplementary Fig. S4). This contains the symbols, the states, the labels and the states that share emission probabilities.
4. set up the configuration file in a text editor. This file contains all the training and testing options.

After defining the model and the configuration file and depending on the chosen options, parameter estimation and testing can be performed in one step. JUCHMME offers the option for self-consistency test, independent test, *k*-fold cross-validation or jackknife procedures. The different model types, training and testing options can be combined efficiently. Various measures of accuracy and reliability can be requested through the configuration file. The trained model can also be used in a stand-alone application.

Finally, after the model is trained, new data may emerge and an update may be possible. Thus, one can easily modify the architecture of existing models. In this way, the HMMpTM model (Tsaousis *et al.*, 2014) was created by modifying the architecture of the HMMTM model (Bagos *et al.*, 2006). This was also the case regarding CW-PRED and CW-PRED2 (Fimereli *et al.*, 2012; Litou *et al.*, 2008).

4 Results and conclusion

The JUCHMME toolkit is implemented in Java and it is publicly available under the GNU license. The software has been under development over the last years and has been used in numerous applications in protein sequence analysis, including prediction of alpha-helical membrane proteins (Bagos *et al.*, 2006), prediction of transmembrane beta-barrels (Bagos *et al.*, 2004a, b; Tsirigos *et al.*, 2016), prediction of lipoprotein signal peptides (Bagos *et al.*, 2008), prediction of signal peptides in Archaea (Bagos *et al.*, 2009), prediction of twin-arginine signal peptides in bacteria (Bagos *et al.*, 2010), joint prediction of transmembrane topology and post-translational modifications (Tsaousis *et al.*, 2014) and prediction of cell-wall anchored proteins in bacteria (Litou *et al.*, 2008). These methods (which we now make available through JUCHMME) could not have been developed easily without the use of this tool, since this would require that the different sub-models corresponding to labels had to be trained separately and then the sub-models had to be combined with arbitrary transitions. The CHMM approach which requires labels is not only more elegant, but also offers the CML methods with the associated algorithms that can efficiently train such models.

The software can be used for building standard HMMs, CHMMs with labeled sequences, or HNNs. JUCHMME supports a versatile architecture in which the models can be freely parameterized in terms of their architecture (i.e. states and transitions between them), the alphabet of the symbols used and the number of labels and sharing of emission probabilities (parameter tying).

A comparison of JUCHMME against the other available tools, such as modhmm (Viklund, 2003), MAMOT (Schütz and Delorenzi, 2008), HMMConverter (Lam and Meyer, 2009), HMMoC (Lunter, 2007) and StochHMM (Lott and Korf, 2014) is provided in Table 1. In brief, JUCHMME implements the widest range of training and decoding algorithms for HMMs. Notably, it is among the few implementations that can handle CHMMs trained with ML or CML, using algorithms for fast and robust convergence and it is, to our knowledge, the only publicly available implementation of HNNs and semi-supervised learning algorithms for HMMs. Contrary to other tools, JUCHMME is user-friendly, since it allows full parameterization through a configuration file, without requiring programming skills. Despite been written in JAVA, JUCHMME is quite fast, especially concerning large models and multicore parallelization increases further the speed (see Supplementary Material). Thus, JUCHMME can be used for original research as well as for educational purposes. HMMs are subjects of active research in our lab, and thus JUCHMME will be continuously updated. We plan,

Table 1. Comparison of features available in different HMM packages

Features	JUCHMME	PHMM	MAMOT	StochHMM	HMMoC	HMMConverter	Modhmm
Types of models							
Class HMM for Labeled Sequences (CHMM)	✓	✓					✓
Hidden Neural Networks (HNN)	✓						
Higher order emissions or transitions (HOHMM, PHMM, HMMSDO)	✓			✓	✓		
pair-HMMs					✓	✓	
Generalized HMMs (GHMM)					✓	✓	
Inhomogeneous chains					✓		
Continuous emissions				✓			
Silent states				✓			✓
Multiple emissions, Joint emissions, Lexical transitions				✓			
Training Algorithms							
Conditional Maximum Likelihood (CML) training	✓						✓
Gradient-descent training (including RPROP)	✓						
Viterbi Training	✓	✓					
Semi-supervised learning (SSL)	✓						
Linear-memory Baum-Welch training						✓	
Linear-memory Viterbi training						✓	
Decoding Algorithms							
Optimal Accuracy Posterior Decoding	✓						
N-best decoding	✓	✓		✓			
Posterior-Viterbi decoding	✓	✓	✓				
Constrained decoding	✓						
Stochastic decoding				✓			
Linear-memory posterior sampling						✓	
Hirschberg decoding algorithm						✓	
Utilities							
Parameter tying (emission sharing)	✓		✓				
Random Sequences Generator	✓		✓				
Multithreaded parallelization	✓						
Can handle MSAs or profiles							✓
Modular architecture for model design							✓
Other Utilities (jackknife test, k–fold cross-validation, early stopping)	✓						
Documentation	✓			✓	✓	✓	✓

Note: All packages provide functionalities for standard HMMs trained with Baum-Welch algorithm and decoded with Posterior/Viterbi algorithms, so these are not listed.

'✓' indicates support for the particular feature within the application.

among others, to extend JUCHMME in order to be able to handle multiple sequence alignments (MSAs), to include silent states and to develop a graphical editor for facilitating model design.

Acknowledgements

We acknowledge support of this work by the project 'ELIXIR-GR: The Greek Research Infrastructure for Data Management and Analysis in Life Sciences' (MIS 5002780) which is implemented under the Action 'Reinforcement of the Research and Innovation Infrastructure', funded by the Operational Programme 'Competitiveness, Entrepreneurship and Innovation' (NSRF 2014–2020) and co-financed by Greece and the European Union (European Regional Development Fund).

Conflict of Interest: none declared.

References

Bagos,P.G. *et al.* (2004a) Faster gradient descent training of hidden markov models, using individual learning rate adaptation. In: Paliouras,G. and Sakakibara,Y. (eds.) *Grammatical Inference: Algorithms and Applications. ICGI 2004. Lecture Notes in Computer Science*, Vol. 3264. Springer, Berlin, Heidelberg.

Bagos,P.G. *et al.* (2004b) A Hidden Markov Model method, capable of predicting and discriminating β-barrel outer membrane proteins. *BMC Bioinformatics*, **5**, 29.

Bagos,P.G. *et al.* (2006) Algorithms for incorporating prior topological information in HMMs: application to transmembrane proteins. *BMC Bioinformatics*, **7**, 189.

Bagos,P.G. *et al.* (2008) Prediction of lipoprotein signal peptides in Gram-positive bacteria with a hidden Markov model. *J. Proteome Res.*, **7**, 5082–5093.

Bagos,P. *et al.* (2009) Prediction of signal peptides in archaea. *Protein Eng. Des. Select.*, **22**, 27–35.

Bagos,P.G. *et al.* (2010) Combined prediction of Tat and Sec signal peptides with hidden Markov models. *Bioinformatics*, **26**, 2811–2817.

Baldi,P. *et al.* (2000) Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, **16**, 412–424.

Baldi,P. and Chauvin,Y. (1994) Smooth on-line learning algorithms for hidden Markov models. *Neural Comput.*, **6**, 307–318.

Durbin,R. *et al.* (1998) *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge: Cambridge University Press.

Eddy,S.R. (1998) Profile hidden Markov models. *Bioinformatics (Oxford, England)*, **14**, 755–763.

Fariselli,P. *et al.* (2005) A new decoding algorithm for hidden Markov models improves the prediction of the topology of all-beta membrane proteins. *BMC Bioinformatics*, **6**, S12.

Fimereli,D.K. *et al*. (2012) CW-PRED: a HMM-based method for the classification of cell wall-anchored proteins of Gram-positive bacteria. In: *Hellenic Conference on Artificial Intelligence*, pp. 285–290. Springer.

Hughey,R. and Krogh,A. (1996) Hidden Markov models for sequence analysis: extension and analysis of the basic method. *Bioinformatics*, **12**, 95–107.

Juang,B.-H. and Rabiner,L.R. (1990) The segmental K-means algorithm for estimating parameters of hidden Markov models. *IEEE Trans. Acoustics Speech Sign. Process*., **38**, 1639–1641.

Käll,L. *et al*. (2005) An HMM posterior decoder for sequence feature prediction that includes homology information. *Bioinformatics*, **21**, i251–i257.

Krogh,A. (1994) Hidden Markov models for labeled sequences. Pattern recognition, 1994. Vol. 2-Conference B: computer vision & image processing., In: *Proceedings of the 12th IAPR International Conference on IEEE*, pp. 140–144.

Krogh,A. (1997) Two methods for improving performance of an HMM and their application for gene finding, Center for Biological Sequence Analysis. *Phone*, **45**, 4525.

Krogh,A. and Riis,S.K. (1999) Hidden neural networks. *Neural Comput*., **11**, 541–563.

Lam,T.Y. and Meyer,I.M. (2009) HMMCONVERTER 1.0: a toolbox for hidden Markov models. *Nucleic Acids Res*., **37**, e139–e139.

Litou,Z.I. *et al*. (2008) Prediction of cell wall sorting signals in gram-positive bacteria with a hidden markov model: application to complete genomes. *J. Bioinf. Comput. Biol*., **06**, 387–401.

Lott,P.C. and Korf,I. (2014) StochHMM: a flexible hidden Markov model tool and C++ library. *Bioinformatics*, **30**, 1625–1626.

Lunter,G. (2007) HMMoC—a compiler for hidden Markov models. *Bioinformatics*, **23**, 2485–2487.

Melen,K. *et al*. (2003) Reliability measures for membrane protein topology prediction algorithms. *J. Mol. Biol*., **327**, 735–744.

Rabiner,L.R. (1989) A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, **77**, 257–286.

Schütz,F. and Delorenzi,M. (2008) MAMOT: hidden Markov modeling tool. *Bioinformatics*, **24**, 1399–1400.

Tamposis,I.A. *et al*. (2018) Extending hidden Markov models to allow conditioning on previous observations. *J. Bioinf. Comput. Biol*., **16**, 1850019.

Tamposis,I.A. *et al*. (2019) Semi-supervised learning of Hidden Markov Models for biological sequence analysis. *Bioinformatics*, **35**, 2208–2215.

Theodoropoulou,M.C. *et al*. (2017) Viterbi training of Hidden Markov Models for labeled sequences. In: *Joint 25th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB) and 16th European Conference on Computational Biology (ECCB)*.

Tsaousis,G.N. *et al*. (2014) HMMpTM: improving transmembrane protein topology prediction using phosphorylation and glycosylation site prediction. *Biochim. Biophys. Acta*, **1844**, 316–322.

Tsirigos,K.D. *et al*. (2016) PRED-TMBB2: improved topology prediction and detection of beta-barrel outer membrane proteins. *Bioinformatics*, **32**, i665–i671.

Viklund,H. (2003) modhmm: Development of a modular HMM package for biological sequence analysis. MSc Thesis, Stockholm Bioinformatics Center.

Zemla,A. *et al*. (1999) A modified definition of Sov, a segment-based measure for protein secondary structure prediction assessment. *Proteins Struct. Funct. Bioinf*., **34**, 220–223.